

Designing a good low-rate sparse-graph code

Iryna Andriyanova

LTHC, School of Compute&Communication Sciences

EPFL, Switzerland

iryna.andriyanova@epfl.ch

Jean-Pierre Tillich

Equipe-Projet SECRET

INRIA Roquencourt, France

jeanpierre.tillich@inria.fr

Abstract— This paper deals with the design of low-rate sparse-graph codes with linear minimum distance in the blocklength. First, we define a necessary condition that a quite general family of graphical codes has to satisfy in order to have linear minimum distance. The condition is formulated in terms of degree-1 and degree-2 variable nodes and of low-weight codewords of the underlying code, and it generalizes results known for turbo codes [8] and LDPC codes. Then, we present a new ensemble of low-rate codes, which is itself a subclass of TLDP codes [3], [4], and which is designed under this necessary condition. The asymptotic analysis of the ensemble shows that its iterative threshold is close to the Shannon limit. In addition to the linear minimum distance property, it has a simple structure and enjoys a low decoding complexity and a fast convergence.

I. INTRODUCTION

Low rate codes play a crucial role in communication systems operating in the low signal-to-noise ratio regime, such as power-limited sensor networks, ultra-wideband communications schemes and code-spread CDMA systems. More recently, it has also been found out that powerful low-rate codes with a fast decoding algorithm can be used in the reconciliation phase of continuous-variable quantum key distribution protocols and allow to increase significantly the range of the protocol [18].

Since the invention of turbo codes [7], a lot of effort was put into designing sparse-graph codes for various applications. This is due to nice features of the iterative decoding algorithm which is used in such codes, namely its low decoding complexity and good performance. But, although the design of good low-rate sparse-graph codes is of great interest, it is not straightforward. By a good low-rate code ensemble we mean an ensemble with iterative decoding threshold close to the channel capacity and a good minimum distance, which is necessary to obtain a low error floor. The problem lies in the fact that, in order to design a low rate code with performance close to the channel capacity, it seems crucial to have a large fraction of variable nodes of degrees 1 and 2 in the code structure (we are going to elaborate on this in Section III). But the presence of a large number of variable nodes of low degrees is not favorable for the minimum distance growth. It may become logarithmic or, even worse, constant. This phenomenon has been quantified in several papers, such as for instance in [20], [?], [19]. A way to circumvent the problem is to introduce some structure in the bipartite graph of a low-rate ensemble, preventing the formation of low-weight codewords.

Recently, some high-performance low-rate schemes have been proposed. A rate-1/10 multi-edge LDPC ensemble with

the threshold -1.09 dB on the AWGN channel was presented in [22]. This construction can be viewed as a serial concatenation of a (3,15) LDPC code and of an LT code and it possesses a complex structure. Its minimum distance growth inherits the minimum distance property of the underlying (3,15) LDPC inner code, i.e. it is linear in blocklength. In [11], authors introduced low-rate ARA-type LDPC codes of different rates in the range from 1/3 to 1/10. The proposed ensembles have iterative thresholds close to the channel capacity and a simpler structure, compared to the previous multi-edge ensemble, but their minimum distance grows polynomially in the blocklength¹. Also, in [17], authors presented a parallel concatenation of Zigzag-Hadamard (ZH) codes. These codes are decoded in a turbo-like fashion, by using the fast Hadamard transform for small Hadamard component codes. This yields a rather low complexity decoding algorithm. The concatenated ZH ensembles have rates down to 0.00105. What concerns their minimum distance properties, the reasoning from [27] can be adapted to show that the minimum distance of such a construction is of order $n^{(M-1)/M}$, where n is the blocklength and M is the number of component ZH codes. This case is treated in [5].

In this work, we propose an alternative low-rate code structure, which enjoys good minimum distance, a good iterative threshold behavior and a low decoding complexity. Our approach avoids to fix a complex bipartite graph structure and enables to get a flexible irregular construction. The point is that the degree distribution of this construction can be optimized by a simple one-dimensional optimization. The procedure that we adapt is the following:

- a) we first provide a necessary condition to ensure linear minimum distance,
- b) then we design a low-rate code ensemble which satisfies this condition based on a component code that enjoys a low-complexity decoding algorithm.

To fulfill the first point, we define a special graph, called the *graph of codewords of partial weight 2*. This graph is derived from connections of variable nodes of degrees 1 and 2 and of low-weight codewords of component codes. In some sense it is a generalization of the subgraph induced by degree-2 variable nodes for LDPC codes [10] to any sparse-graph code ensemble.

Tail-biting Trellis LDPC (or TLDP) codes have been introduced in [3], [4]. This family enjoys an iterative threshold

¹more precisely, it is of order $O(n^{3/4})$, see [5]

close to the channel capacity, a linear minimum distance and a very low decoding complexity. Examples of TLDPC codes of rates $1/3$ and $1/2$ were presented in [3], [4]. In this paper, we utilize the framework of TLDPC codes to design a code ensemble of lower rate. We propose a new TLDPC component code, having a very simple structure and, therefore, a low-complexity decoding algorithm. Moreover, the proposed component code has another interesting feature, which makes the obtaining of linear minimum distance possible: the supports of its low-weight codewords are distributed among the code positions in such a way that the union of intersecting supports form *disjoint* clusters. We will discuss this property in details later on in the paper. We also emphasize that our choice of the component code allows to have a large non-zero fraction of degree-1 variable nodes in the code structure, while keeping the minimum distance grow linearly in the blocklength.

To design a low-rate TLDPC ensemble both with linear minimum distance and an iterative decoding threshold close to the channel capacity, we put a constraint on the maximum allowed fraction of degree-2 variable nodes and optimize over the degree distribution of variable nodes by using EXIT charts. Moreover, in order to satisfy the necessary condition for linear minimum distance we have found, we propose a structured way to generate the permutation for edges connected to degree-2 variable nodes. There is no other constraint on the generation of the permutation for other edges in the bipartite graph, it is assumed to be drawn uniformly at random.

The paper is organized as follows. In the next section the graph of codewords of partial weight 2 and a necessary condition for linear minimum distance are provided. Then we discuss in Section III why it is important to put degree one variable nodes in the graphical structure. A general introduction to TLDPC codes and a presentation of the new low-rate ensemble are given in Section IV. Numerical results are shown in Section V. Section VI contains some discussion on the topic.

II. NECESSARY CONDITION FOR LINEAR MINIMUM DISTANCE

The goal of this section is to formulate a necessary condition for linear minimum distance that could be applied to any sparse-graph code ensemble. We construct a *graph of codewords of partial weight 2* and consider cycles in it. It happens that one can define a logarithmic upper bound on the minimum distance in terms of these cycles. The upper bound can be equivalently expressed in terms of the average degree of the graph of codewords of partial weight 2. This result leads us to the aforementioned necessary condition.

A. Common Representation for Sparse-Graph Codes

For the sake of generality, we use the following general representation for all sparse-graph codes that was first described in [26]:

Definition 1 (Common construction and base code): The construction produces a binary code of length n with the help of two ingredients:

- (i) a binary code \mathcal{B} of length m , with $m \geq n$. This code is called the *base code*;
- (ii) a bipartite graph between two sets V and W of vertices of size n and m respectively, where the degree of any vertex in W is exactly one and the degree of the vertices in V is specified by a degree distribution $\Lambda = (\lambda_1, \lambda_2, \dots, \lambda_s)$ where λ_i denotes the fraction of edges of the bipartite graph which are incident to vertices of V of degree i .

The bipartite graph together with the base code specifies a code of length n as the set of binary assignments of V such that the induced assignments² of vertices of W belong to \mathcal{C} . If we denote the rate of the base code by R_b , then it is straightforward to check that the rate of the code obtained by this construction is at least equal to the *designed rate* R which is given here by the expression

$$R \stackrel{\text{def}}{=} 1 - (1 - R_b)\bar{\lambda}$$

where $\bar{\lambda}$ is the average left degree, which is given by

$$\bar{\lambda} \stackrel{\text{def}}{=} \frac{m}{n} = \frac{1}{\sum_i \frac{\lambda_i}{i}}.$$

It is common to present the degree distribution Λ in its polynomial form:

$$\Lambda(x) = \sum_{i=1}^s \lambda_i x^{i-1}.$$

Most sparse-graph code constructions such as LDPC codes or parallel turbo-codes can be viewed as a particular instance of this construction.

Example [LDPC codes] Classical LDPC codes are an example of unstructured code ensembles. The LDPC base code is the juxtaposition of parity codes. $\Lambda(x)$ is the left degree distribution of the LDPC code ensemble. \diamond

When the bipartite graph between sets V and W has some special structure, we say that it is a *structured* code ensemble.

Example [Parallel turbo codes] Parallel turbo codes are one of the first structured ensembles which have been suggested in the literature. The base code of a parallel turbo ensemble is the juxtaposition of several convolutional codes, the positions of which are divided into two subsets, the first one is formed by the information bits and the second one by the redundancy bits. The sets V and W in the bipartite graph are also divided into two subsets, the subset of information nodes and the subset of redundancy nodes. A node in V and a node in W can be connected only if they belong to the same subset type and redundancy nodes have all degree one. \diamond

The standard decoding procedure [23] for sparse-graph codes is the following. At each decoding iteration, base code decoding is performed in order to get extrinsic messages for bits of the base code, then intrinsic messages at the variable node side are calculated. At the end of decoding, after some number of iterations, a posteriori messages of code bits are computed. The decoding complexity therefore depends on

²a vertex in W receives the same assignment as the vertex in V it is connected to.

the complexity of the base code decoding, on the degree distribution of variable nodes (the higher the node degree the more complex decoding gets) and on the number of decoding iterations which are needed to be performed (i.e. the decoding convergence speed).

B. Graph of Codewords of Partial Weight 2

A position in the base code C is said to have degree i if it is connected to a node of degree i in V . Notice that in the graphical representation we allow variable nodes to be of degree ≥ 1 and, therefore, we allow positions of degree 1 in C . The location of these positions has a crucial impact on the minimum distance of the overall code, which may even become constant if there is a codeword in C whose support contains only positions of degree 1. In what follows, this case is supposed to be avoided.

To study the minimum distance behavior, we define the notion of codewords of the base code of partial weight 2, which will lead us to the notion of clusters.

Definition 2 (Codewords of C of partial weight 2):

Codewords of C of partial weight 2 are the codewords that involve exactly two non-zero positions of degree > 1 . Thus, all other non-zero positions of them have degree 1.

Definition 3 (Clusters): A cluster is an ensemble of positions of degree > 1 in C , so that for any two positions i and j from this ensemble there exists a codeword of partial degree 2 in C containing i and j .

The simplest example of clusters can be given in the case of LDPC codes.

Example [Standard LDPC codes, $\lambda_1 = 0$] Remember that the base code is a juxtaposition of small parity codes. Any two positions of the same parity code form the support for one codeword of partial weight 2. Thus, clusters correspond to ensembles of positions belonging to the same parity codes. \diamond

With this notion of cluster, we can define now the graph of codewords of partial weight 2:

Definition 4 (Graph of codewords of partial weight 2):

The graph of codewords of partial weight 2 is a graph $G = (\tilde{V}, E)$ with vertex set \tilde{V} and edge set E . V is equal to the set of clusters and there is an edge e_{ij} between two clusters \tilde{v}_i and \tilde{v}_j iff there exist two positions x_k and x_l of the base code, belonging to the clusters \tilde{v}_i and \tilde{v}_j respectively, which join the same degree-2 variable node.

Example [Standard LDPC codes] Continuing the previous example, the graph of codewords of partial weight 2 for a LDPC code contains clusters that correspond to parity checks in the code structure. Two clusters are connected if their corresponding parity checks are connected through a degree-2 variable node in the bipartite graph of the code. \diamond

C. Cycles in the Graph of Codewords of Weight 2 and Its Average Degree

It is well known [9] that the first source of low weight codewords when an LDPC code is chosen at random are cycles in the Tanner graphs containing only variable nodes of degree

2. Note that these are in one-to-one correspondence with cycles in the graph of codewords of partial weight 2. It turns out that cycles in this graph are in general for sparse-graph codes the first cause of problems for the minimum distance. Our necessary condition for having a linear minimum distance roughly says that there should be no cycles of sublinear length in this graph. In order to state this condition, we need two more definitions, those of node weights and of cycle weights of G :

Definition 5 (Node weight): For a node v in \tilde{V} and two edges i and j connected to it, we define a node weight $w_{i,j}^v$ as follows. By the very definition of a cluster and of the graph of codewords of partial weight 2, these two edges correspond to two positions of degree 2 and they form together with a certain number a of positions of degree 1 the support of a codeword of partial weight 2. We let $w_{i,j}^v$ be equal to this number a .

Definition 6 (Cycle weight): The weight l of a cycle $C = (V_C, E_C)$ in G is equal to

$$l = |E_C| + \sum_{v \in V_C} w^v,$$

where w^v is the node weight associated with vertex v in V_C and the two edges in E_C connected to v .

There is a fundamental correspondence between cycles in the graph of codewords of partial weight 2 and low-weight codewords of C :

Proposition 1: A cycle of weight l in the graph of codewords of partial weight 2 induces a codeword of weight l in the sparse-graph code.

Proof. If $C = (V_C, E_C)$ is a cycle in G , we associate to it a configuration $\mathbf{x} = (x_1, x_2, \dots, x_m)$ of positions of the base code in which

- positions of the base code of degree 2 are set to 1 if in the Tanner graph they are connected to the variable nodes of degree 2 that are associated with edges in E_C ;
- a set B of positions of degree 1 is set to 1 if they form a codeword of the base code of partial weight 2 with two corresponding positions of degree 2;
- all other positions in \mathbf{x} are set to 0.

Denote by w^v the size of the set B for a node $v \in V_C$. The point is that the configuration \mathbf{x} is obviously a codeword of the base code. It has weight $2|E_C| + \sum_{v \in V_C} w^v$. $2|E_C|$ non-zero bits of \mathbf{x} are connected to degree-2 variable nodes and the rest of them is connected to degree-1 variable nodes. Thus, there are $|E_C| + \sum_{v \in V_C} w^v$ variable nodes participating in the configuration \mathbf{x} , and they correspond to a codeword of weight $|E_C| + \sum_{v \in V_C} w^v$. \square

Notice that the weight of the smallest cycle in the graph of codewords of partial weight 2 is an upper bound on the code minimum distance. Therefore:

Corollary 1: If all the node weights $w_{i,j}^v$ of a given graph of codewords of partial weight 2 are smaller than some constant $a > 0$, $a \in \mathbb{N}$, then the minimum distance of its corresponding sparse-graph code is upper bounded by $(a + 1)|E_C|$.

Corollary 2: If a given graph of codewords of partial weight 2 contains a cycle of logarithmic weight, the minimum

distance of the sparse-graph code ensemble is logarithmic in the blocklength.

Corollary 2 can be equivalently expressed in terms of the average degree of the graph of codewords of partial weight 2.

Theorem 1 (Upper bound on d_{\min}): Consider a sparse-graph code ensemble for which the graphs of codewords of partial weight 2 have node weights upper bounded by a small positive integer a . If all the average degrees of these graphs is greater than $2 + \epsilon$ for some $\epsilon > 0$, then the minimum distance of the ensemble grows logarithmically in the blocklength.

Proof. Consider a sparse graph code in this ensemble. Let G be the associated graph of codewords of partial weight 2 and d_{\min} be the minimum distance of the code. Let g be the girth of G and Δ be its average degree. By Corollary 1 we know that $d_{\min} \leq (a + 1)g$. To upperbound this last quantity we use the Moore bound for irregular graphs [1] which asserts that the number of vertices n of G satisfies the following inequality

$$n \geq 2 \frac{(\Delta - 1)^t - 1}{\Delta - 2}$$

where $t = \lfloor \frac{g}{2} \rfloor$. This implies

$$t \leq \log_{\Delta-1} \left(\frac{\Delta - 2}{2} n + 1 \right)$$

We now conclude by:

$$\begin{aligned} d_{\min} &\leq (a + 1)g \\ &\leq (a + 1)(2t + 1) \\ &\leq (a + 1) \left(2 \log_{\Delta-1} \left(\frac{\Delta - 2}{2} n + 1 \right) + 1 \right). \end{aligned}$$

□

D. Necessary Condition

Now, the following necessary condition follows immediately:

While constructing a sparse-graph code ensemble with a linear growth of the average minimum distance, cycles of sublinear weights in the corresponding graph of codewords of partial weight 2 must be avoided. Or, equivalently, the average degree Δ of the graph of codewords of partial weight 2 must be smaller than or equal to 2.

This necessary takes the following form for LDPC code ensembles.

Example [LDPC codes] Consider a (possibly) irregular LDPC code ensemble. Let λ_2 be the fraction of its degree-2 variable nodes and let ρ be the average degree of its check nodes (it is equal to the fraction $\frac{m}{r}$, where r is the number of check nodes of the Tanner graph and m its number of edges). The number of clusters is equal to this number of check nodes $r = \frac{m}{\rho}$ (see the previous example about standard LDPC code ensembles). The average degree of this graph should be upper-bounded by 2, and therefore the graph of codewords of partial weight 2 should not have more than r edges. This means that there should be at most r variable nodes of degree 2 in the

graph. This number is equal to $\frac{\lambda_2 m}{2}$. This eventually implies that

$$\frac{\lambda_2 m}{2} \leq r = \frac{m}{\rho}.$$

In other words, if we want the minimum distance to grow linearly in the blocklength, the condition $\lambda_2 \rho \leq 2$ should be verified. ◊

Notice that $\Delta = 2$ is the critical case. It corresponds to the situation when G contains one or several cycles of linear length. It has been shown in [27] that for LDPC codes and $\Delta = 2$, the minimum distance is polynomial in the blocklength. For more general families of sparse graph-codes (by adding for instance state nodes in the standard construction of LDPC codes) this is not true anymore as shown by the example of Section V in [19] where the minimum distance is linear.

Notice that until now were dealing with sparse graph codes ensembles with bounded node weights. For some code ensembles the node weights are unbounded, as it is the case for turbo codes. With a little work, our results can be still extended to unbounded weights and, therefore, Corollary 2 and Theorem 1 will hold. For completeness of the demonstration, we elaborate the bound for parallel turbo codes, which leads to a much shorter proof of the result obtained by Breiling [8]

Theorem 2 ([8]): The minimum distance of parallel turbo codes with two parallel components grows at most logarithmically in blocklength.

Proof. Let us construct the graph of codewords of partial weight 2 for parallel turbo codes. For simplicity, assume that both component encoders are recursive systematic convolutional encoders of type $(n, 1)$ and that they are equal. Then, there exists t such that for any information position i in the convolutional code there is a codeword of partial weight 2 with information support $\{i, i + t\}$ and with redundancy weight w . Other codewords of partial weight 2 are deduced by addition. They have information support $\{i, i + kt\}$, their redundancy weight is at most kw and they all belong to the same cluster in G . Therefore, G consists of (at most) $2t$ clusters³, which are connected through N edges, where N is the number of information bits in the turbo code.

Notice that the node weights of the clusters are unbounded. To circumvent this difficulty, we form smaller clusters by partitioning each cluster into subclusters of size 3 of the form $\{i, i + t, i + 2t\}$. We obtain a new graph of codewords of partial weight 2, denoted by G' , with $2N/3 + O(1)$ clusters and of degree 3. Moreover, the node weights of G' are bounded by $2w$. Therefore, G' has a cycle of size at most $2 \log_2(2N/3 + O(1))$ and of weight at most $2(1 + 2w) \log_2(2N/3 + O(1))$. This yields a codeword of weight $2(1 + 2w) \log_2(2N/3 + O(1))$ in the turbo code by Proposition 1. □

³The factor 2 comes from the fact that there are two convolutional codes each one coming with its own set of clusters.

III. ON THE USEFULNESS OF DESIGNING SPARSE GRAPH CODES WITH DEGREE ONE NODES

It might be worthwhile to quote [23] here: “Given the importance of degree-two edges, it is natural to conjecture that degree-one edges could bring further benefits”. This can be illustrated by the following phenomenon : it has been repeatedly observed that in general turbo-codes require far less iterations to be decoded than LDPC codes. While this is not always true, for instance LDPC codes where all parity-checks involve at least two bits of degree 2, can be decoded in a turbo-like fashion (or more precisely in a convolutional fashion). This seems to strongly decrease in this case the required number of iterations for completing iterative decoding. It has also been observed many times that turbo-codes tend to outperform LDPC codes at shorter block-length. A possible explanation for the better behavior of turbo-codes compared to LDPC codes could be given by the fact that the former are decoded with a graphical structure with bits of degree 1 (which are given by the redundancy bits) which are absent in the case of LDPC codes. This is also corroborated by the fact that a small modification of LDPC codes which allows for bits of degree 1 can produce LDPC codes with a much steeper waterfall region than conventional LDPC codes [21, Rate $\frac{1}{2}$ example in Table VIII].

Obtaining sparse graph codes with a steep waterfall region and requiring a moderate amount of iterations is quite problematic in the case of low-rate codes. Conventional LDPC codes are well known for being poor in the low rate regime. Again, the first example of a modified LDPC code ensemble with good iterative decoding performance at low rates is given by a structure with bits of degree 1 [21, Rate $\frac{1}{10}$ example of Table X].

One of the purpose of this section is to give a heuristic explanation of all these phenomena (less iterations for turbo-codes or LDPC codes decoded in a turbo-like fashion, better performance at short block-length for turbo-codes). The idea we wish to convey here is that this is a consequence of the presence of bits of degree 1 (or the presence of “hidden” bits of degree 1 in the case of LDPC codes decoded in a turbo-like manner).

The heuristic explanation we will provide will be with the help of EXIT charts on the binary erasure channel. The same kind of explanation could also be given for other channels (by some hand-waving by asserting that the fundamental relation, namely Theorem 1, which holds for the binary erasure channel, holds approximately for other channels). It is well known that in this case the EXIT chart predicts accurately the infinite length behaviour of the code, and that they represent in some sense the “average” trajectories for finite length : they are given by horizontal and vertical steps between both curves. Iterative decoding is typically succesful (meaning that it is successful with probability tending to 1 as the length goes to infinity) if and only if the EXIT chart of the variable nodes is above the EXIT chart of the variable nodes. The area $\Delta\mathcal{A}$ between both curves has a very nice interpretation : it is linked

with the distance to capacity. It has basically been observed in [6] (generalising a result first proved by Shokrollahi on LDPC codes [24]) that in order to have a capacity achieving sequence of codes (in the sense of [24]) the area between the two EXIT charts of a code in the sequence should go to zero.

To be more specific, there are two EXIT charts that we will consider here:

- 1) the *EXIT chart of the variable nodes* which is the curve which expresses how the average entropy of code positions given by their incoming extrinsic probabilities behaves in term of the average entropy of the code positions once their intrinsic probabilities have been computed at the variable node level (here the average is taken only over variable nodes of degree 2). When there are no variable nodes of degree 1 and for an erasure channel of probability p , this curve is given by the set of points $(p\lambda(x), x)$ where x ranges over $[0, 1]$. When there are variable nodes of degree 1 this curve is given by the set of points $(p \frac{\sum_{i>1} \lambda_i x^{i-1}}{\sum_{i>1} \lambda_i}, x)$. In other words this is the set of points $\left\{ \left(\frac{p(\lambda(x) - \lambda_1)}{1 - \lambda_1}, x \right), x \in [0, 1] \right\}$. If we bring the degree distribution of the edges of left degree greater than 1,

$$\tilde{\lambda}_i \stackrel{\text{def}}{=} \frac{\lambda_i}{1 - \lambda_1} \quad (1)$$

for $i > 1$ (and $\tilde{\lambda}_1 = 0$) and the associated polynomial

$$\Lambda(\tilde{x}) = \sum_{i>1} \tilde{\lambda}_i x^{i-1} = \sum_{i>1} \frac{\lambda_i}{1 - \lambda_1} x^{i-1}, \quad (2)$$

then the EXIT chart for the variable nodes is given by the curve $\left\{ (p\Lambda(\tilde{x}), x), x \in [0, 1] \right\}$.

- 2) The *EXIT chart of the base code* which is the curve which expresses how the average entropy of code positions given by their outgoing extrinsic probabilities computed after decoding the base code behaves in term of the average entropy of the code positions before decoding. When the base code consists in a juxtaposition of single parity-check codes of length r (which corresponds to the right-regular LDPC case) this curve is given by the set of points $\left\{ (x, 1 - (1 - x)^{r-1}), x \in [0, 1] \right\}$.

Iterative decoding converges for an infinite length code if and only the base code EXIT curve lies below the EXIT chart of the variable nodes. The statement we are going to give below is not really stated in [6] but is in essence only a corollary of the results given in this article.

Theorem 1: [Area theorem] Let $\Delta\mathcal{A}$ be the area between the two aforementioned EXIT charts.

$$\Delta\mathcal{A} = \frac{C(p) - R}{\tilde{\lambda}(1 - \lambda_1)}$$

where $C(p)$ is the capacity of the binary erasure channel with probability p , that is $C(p) = 1 - p$.

This result raises several comments.

- For a same gap to capacity and fixed $\tilde{\lambda}$, the area between the EXIT charts of the variable nodes and the one of

the base code is larger in the presence of degree one nodes than without by a factor of $\frac{1}{1-\lambda_1}$. This can be quite significant for a large proportion of degree one nodes.

- It is not necessarily true that the number of iterations is smaller when the area between both EXIT charts is larger (this also depends on the shapes of both curves). However, this is a strong indication that both curves are further apart with degree one nodes, and thus that this tends to decrease the number of iterations for iterative decoding. Note that turbo-codes, and especially low rate turbo-codes have a large number of degree one nodes, and that this might well be the explanation for the small amount of iterations needed to decode them in comparison to LDPC codes decoded by the standard Gallager algorithm (the latter have no degree one nodes at all).
- This widening between both curves has also a positive influence on the slope in the waterfall region as was put forward in [14], [15], [16] (see also [2] for a rigorous (and somewhat corrected) derivation of the exponential behavior of the probability of error after decoding suggested in the aforementioned references in the case of the binary erasure channel. For a possible generalization of the formulas obtained in [2] to more general channels see [13], [12]). This might also be the explanation why turbo-codes are believed to be better for moderate lengths than LDPC codes. In this case, it is essential to have a steep waterfall region.

This theorem is obtained through several considerations on the EXIT chart of the variable nodes and the base code. First of all, recall [6] that the area under the EXIT chart for the variable nodes is given by

Proposition 1:

$$\mathcal{A} = 1 - p \frac{\frac{1}{\bar{\lambda}} - \lambda_1}{1 - \lambda_1}$$

Proof. Recall that the EXIT chart for variable nodes on the binary erasure channel is given by the set of points $(\frac{1}{1-\lambda_1}p(\lambda(x) - \lambda_1), x)$ where x ranges over $[0, 1]$. The area below this curve is given by

$$\begin{aligned} \mathcal{A} &= 1 - \int_0^1 \frac{p(\lambda(x) - \lambda_1)}{1 - \lambda_1} dx \\ &= 1 - \frac{p \sum_{i>1} \frac{\lambda_i}{i}}{1 - \lambda_1} \\ &= 1 - p \frac{\sum_i \frac{\lambda_i}{i} - \lambda_1}{1 - \lambda_1} \\ &= 1 - p \frac{\frac{1}{\bar{\lambda}} - \lambda_1}{1 - \lambda_1} \end{aligned}$$

□

The area below the EXIT chart of the base code is given by the following proposition which is a simple corollary of [6, Theorem 1]

Proposition 2: Assume that the bits of degree 1 of the base code \mathcal{B} can be completed to form an information set for \mathcal{B} .

Then the area \mathcal{A} under the EXIT chart of the base code over the binary erasure channel is given by

$$\mathcal{A} = \frac{R_b - (1-p)\lambda_1}{1 - \lambda_1}$$

where R_b denotes the rate of the base code.

Proof. From Theorem 1 in [6] we know that

$$\mathcal{A} = \frac{H(V|Y)}{(1 - \lambda_i)m}$$

Here V consists of a codeword of the base code which is chosen uniformly at random and Y is the transmitted codeword where all positions of degree > 1 have been erased and all positions of degree 1 have been erased with probability p . Let Z be the number of non-erased positions of V . Note that

$$H(V|Z = t) = R_b m - t$$

(this is a consequence on the assumption made on the positions of degree 1). From this we deduce that

$$H(V|Y) = R_b m - (1-p)\lambda_1 m$$

and the proposition follows immediately. □

We are ready now for the proof of Theorem 1

Proof of Theorem 1.

By proposition 2 and 1 we have (as long as the EXIT chart of the base code lies below the EXIT chart of the variable nodes)

$$\begin{aligned} \Delta \mathcal{A} &= 1 - p \frac{\frac{1}{\bar{\lambda}} - \lambda_1}{1 - \lambda_1} - \frac{R_b - (1-p)\lambda_1}{1 - \lambda_1} \\ &= \frac{\bar{\lambda}(1 - \lambda_1) - p(1 - \lambda_1\bar{\lambda}) - R_b\bar{\lambda} + (1-p)\lambda_1\bar{\lambda}}{\bar{\lambda}(1 - \lambda_1)} \\ &= \frac{(1 - R_b)\bar{\lambda} - p}{\bar{\lambda}(1 - \lambda_1)} \\ &= \frac{C(p) - R}{\bar{\lambda}(1 - \lambda_1)} \end{aligned}$$

□

A straightforward corollary of Theorem 1 is

Corollary 3:

$$\frac{d\Delta \mathcal{A}}{dp} = \frac{1}{\bar{\lambda}(1 - \lambda_1)}$$

To illustrate these facts let us consider a particular sparse graph code family obtained by the TLDPC construction which will be detailed in Section IV. The code obtained by this construction is of rate $\frac{1}{10}$. It is obtained by choosing a base code of rate $\frac{1}{2}$ which has a tail-biting trellis with only binary states and a bipartite graph with degree distribution

$$\Lambda(x) = \frac{1}{3} + 0.296x + 0.117x^2 + 0.006x^3 + 0.114x^4 + 0.134x^{11}.$$

In other words, there is a fraction $\lambda_1 = \frac{1}{3}$ of degree 1 edges here. This code family is almost capacity achieving for the erasure channel, where it can correct up to an erasure

probability of $p_0 = 0.896$ as shown by the EXIT charts of the base code and the variable nodes for this probability given by the red and blue solid lines in Figure 1. Note that these two curves almost coincide as is the case for almost capacity achieving ensembles.

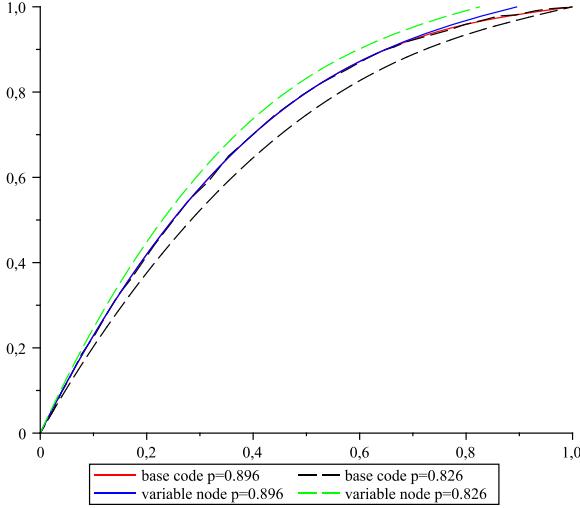


Fig. 1. EXIT chart of a TLDP code of a rate $\frac{1}{10}$ code with $\lambda_1 = \frac{1}{3}$

We also observe in this figure the EXIT charts obtained for an erasure probability which is below the threshold, that is $p = p_0 - \Delta p$ with $\Delta p = 0.07$. The two EXIT charts given by black and green dashed lines are of course much further apart. Performing the same optimization for LDPC codes yields a completely different result. We have chosen an LDPC code with right degree distribution

$$\rho(x) = \frac{1}{10}x + \frac{1}{2}x^2 + \frac{2}{5}x^3,$$

meaning that a fraction $\frac{1}{10}$ of the edges of the bipartite graph go to a check node of degree 2, half of the edges go to a check node of degree 3 and the remaining edges go to a check node of degree 4. The reason of this choice is to obtain a base code whose EXIT chart has about the same shape as the EXIT curve of the base code of the TLDP construction as shown in Figure 2. This allows a fair comparison of both ensembles. Performing a degree optimization over the variable nodes yields (with the same constraint as in the TLDP code, namely having a maximal left degree of 12) yields the following degree distribution

$$\Lambda(x) = 0.486x + 0.165x^2 + 0.037x^3 + 0.15x^4 + 0.132x^{10} + 0.03x^{11}$$

The rate of this ensemble is almost $\frac{1}{10}$ and the erasure probability it is able to sustain is slightly less than for the TLDP code : the noise threshold p_0 satisfies $p_0 \approx 0.8933$. We have plotted in Figure 3 the EXIT curves of the base code and the EXIT curves of the variable nodes at the threshold p_0 and also for $p = p_0 - \Delta p$ with $\Delta p = 0.07$.

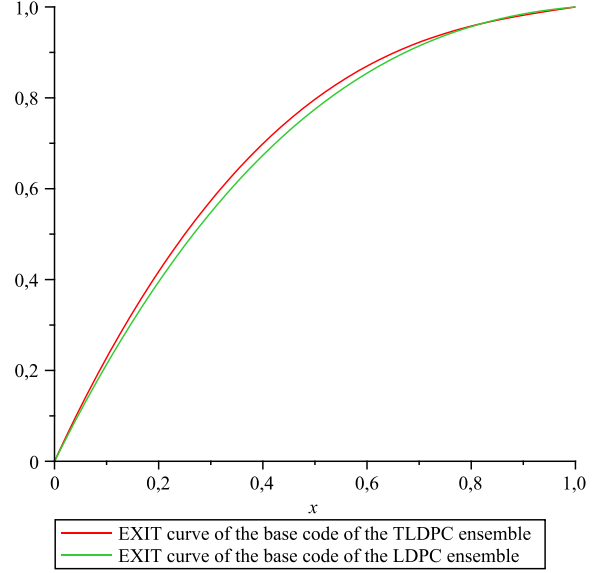


Fig. 2. EXIT charts of both base codes

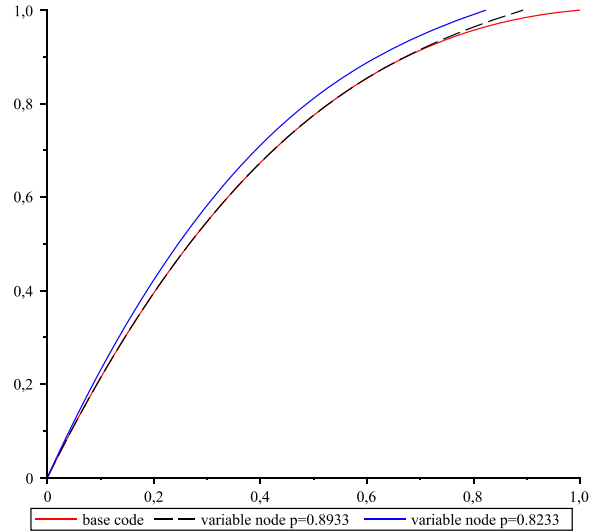


Fig. 3. EXIT chart of an LDPC code of rate $\frac{1}{10}$ code.

It turns out that in the LDPC case the EXIT curves of the base code and of variable nodes are much closer at $p = p_0 - \Delta p$ than they are in the TLDP case. This can be explained by the fact that in the LDPC case the EXIT curve of the base code does not change when channel conditions improve : it is always given by the graph of the function $x \mapsto 1 - \rho(1 - x)$. This is not the case anymore when there are bits of degree 1. In this case, when the channel conditions improve, the new EXIT curve of the base code moves away from the EXIT curve of the base code obtained at the threshold p_0 . The area gained

here is quantified by Proposition 2 and the area $\Delta\mathcal{A}_1$ between the EXIT charts of the base code at p_0 and $p_0 - \Delta p$ is given by

$$\Delta\mathcal{A}_1 = \frac{\lambda_1}{1 - \lambda_1} \Delta p.$$

This area is illustrated for the aforementioned TLDPCC code of rate $\frac{1}{10}$ in Figure 4 by the area colored in blue. This really

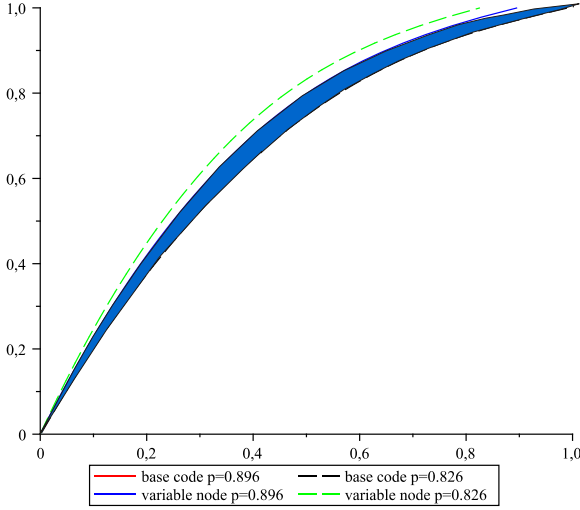


Fig. 4. a figure showing $\Delta\mathcal{A}_1$ for the TLDPCC code of rate $\frac{1}{10}$.

accounts for the difference between the TLDPCC case and the LDPC case. This clearly results in a number of iterations needed for convergence which is much smaller when there are bits of degree 1. The fact that the EXIT curves are also further apart is quite likely to improve the slope of the waterfall region. These two facts are essential for obtaining good low-rate codes.

We may also remark that though the formula given in Proposition 1 seems to depend on λ_1 too, this quantity has *no* influence at all on how fast the variable node EXIT charts move away when the channel conditions improve. Indeed, it is easy to check that the area $\Delta\mathcal{A}_2$ between the EXIT charts of the variable nodes at p_0 and at $p_0 - \Delta p$ is given by

$$\begin{aligned} \Delta\mathcal{A}_2 &= \frac{\frac{1}{\tilde{\lambda}} - \lambda_1}{1 - \lambda_1} \Delta p \\ &= \frac{\Delta p}{\tilde{\lambda}}. \end{aligned}$$

where $\tilde{\lambda} \stackrel{\text{def}}{=} \frac{1}{\sum_{i>1} \frac{\lambda_i}{i}}$ and the $\tilde{\lambda}_i$'s form the degree distribution of the variable nodes of degree > 1 (as defined by Equation (1)). It is basically a consequence of the fact that the EXIT chart of the variable node really depends on $\Lambda(\tilde{x})$ (as defined by Equation (2)) and not on $\Lambda(x)$. However, this dependency on $\tilde{\lambda}$ seems to suggest that in order to reduce the number of decoding iterations and improve the slope of the waterfall region one should aim at obtaining sparse graph codes for

which $\tilde{\lambda}$ is as small as possible (ideally $\tilde{\lambda} = 2$). This is precisely what happens for standard parallel turbo-codes for which $\tilde{\lambda}$ is indeed equal to 2. This also provides a heuristic explanation for the common belief that sparse graph codes with a small $\tilde{\lambda}$ should be sought for are good if one needs sparse graph codes with a good iterative decoding behavior for small and moderate lengths (where the slope of the waterfall region is of paramount importance for obtaining good performance). Notice that the case $\tilde{\lambda} = 2$, corresponds to $\Lambda(\tilde{x}) = x$ and therefore the EXIT chart of the variable node curve is given on the erasure channel by the straight line of equation $x = py$. If we want that the corresponding code ensemble is almost capacity achieving this implies that the EXIT chart of the base code should be close to such a straight line. Such a behavior is obtained for the EXIT charts of the base codes chosen for the TLDPCC code families which will be defined in the following section. For instance, for the TLDPCC codes given by families (A) and (B) taken from [4] the EXIT charts of the corresponding base codes look like:

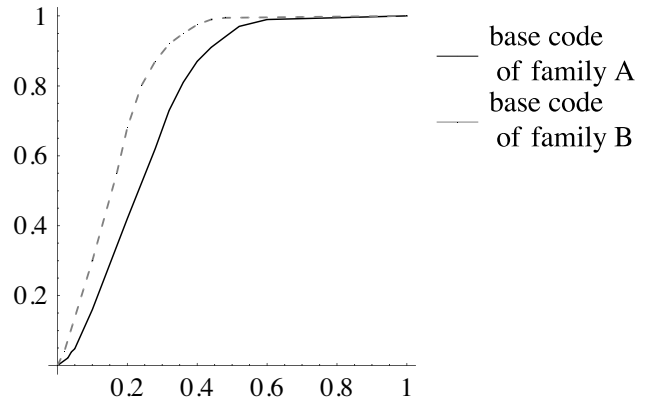


Fig. 5. Exit charts for families (A) and (B).

IV. TLDPCC ENSEMBLE OF RATE 1/10 SATISFYING THE NECESSARY CONDITION ON THE LINEAR MINIMUM DISTANCE

TLDPCC codes form a structured code family which were first proposed in [3] to meet the requirements of a low iterative decoding complexity, a linear minimum distance and an iterative threshold close to the channel capacity. They can be viewed as a slight modification of an LDPC code which allows degree 1 nodes in the structure by adding state nodes to the structure. They differ from the multi-edge approach suggested in [21] in two ways: (i) the base code which is decoded is not a juxtaposition of single parity-check codes but it is a tail-biting convolutional codes with only binary state nodes, (ii) its structure permits a one-dimensional degree optimization rather than having to perform a multi-dimensional optimization as is the case for multi-edge LDPC codes. They

were designed by using several construction methods put together, some of them apply to the base code, some of them deal with the bipartite graph.

A. TLDPC Codes

In this subsection we give the general definition of TLDPC codes and describe the construction methods that were used.

1) *TLDPC base code*: For the moment, suppose that there are no bits of degree 1 in the code structure. Then the TLDPC base code is defined as follows:

Definition 7 (TLDPC base code): The base code of the TLDPC code is a tail-biting convolutional code, the Tanner graph of which is presented in Figure 6. Black vertices are

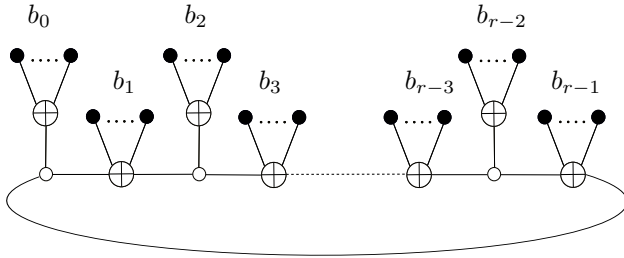
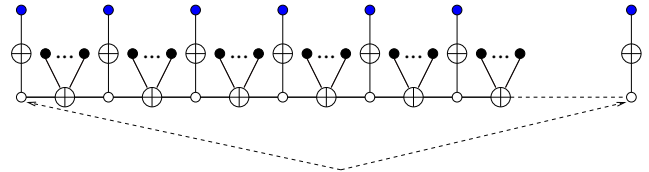


Fig. 6. Tanner graph of a TLDPC base code.

associated with positions of the base code, white vertices with non-transmitted states, and the \oplus 's represent parity-check equations. The first and the last state nodes are identified. The number of black vertices associated to the i -th parity-check node is denoted by b_i .

In the presence of degree-1 bits, the TLDPC base code is defined in a similar matter, yet the positions of degree 1 in the base code have to be specified. It should be noted that in their tail-biting version, systematic RA (Repeat and Accumulate) codes, systematic IRA codes (irregular repeat and accumulate) codes and most of the LDPC codes which are standardized are in fact a subclass of TLDPC codes. We mean here those LDPC codes which have the same amount of degree 2 variable nodes as there are parity-checks and where these parity-check nodes are connected together by a single chain of degree 2 variable nodes. They are decoded as a turbo-code and not as an LDPC code. All these codes are particular TLDPC codes for which all b_i 's are equal to 1 for even values of i (which corresponds to transmitted variable nodes belonging to the parity-checks that involve a single state node), see Figure IV-A.1 and where the corresponding variable nodes are all chosen to be of degree 1. The positions of degree 1 are redundancy bits of the code.

As explained at the end of the previous section, the point of such a construction is that the EXIT curve of the base code is close to be a straight line. This is quite helpful to design codes with require only a moderate number of iterations to converge and also to obtain a rather steep waterfall region. Moreover, the base code is in essence not more complex to decode than single parity-check codes, contrarily to what happens with standard turbo-codes. It may also allow much more degree 2



the two end state nodes are identified

- transmitted variable node of degree 1
- transmitted variable node of degree > 1
- state node

Fig. 7. Base code for systematic (I)RA codes and the aforementioned LDPC codes

variable nodes in the structure than conventional LDPC codes and still form a code family with linear minimum distance. Again, this can be quite beneficial for the speed of iterative decoding convergence and for the waterfall region.

In order to design code ensembles with linear minimum distance, an additional constraint is to be put on the choice of a base code to satisfy the necessary condition given in Section II-D: *the clusters, formed by codewords of partial weight 2 in the designed base code, must have bounded weights*. This condition ensures that the base code has a *linear* number of clusters. This avoids for instance systematic IRA codes where it can be checked that there is a single cluster. In this way, a non-zero fraction of degree-2 variable nodes may be allowed for a linear minimum distance growth.

2) *Structure of the bipartite graph*: A structure on the permutation of the edges connected to degree-2 variable bits in the bipartite graph needs to be put in order to satisfy the necessary condition on linear minimum distance. The permutation for edges connected to variable nodes of degrees > 2 is supposed to be generated uniformly at random.

We began the design of the code ensemble by choosing the base code. Next, we perform the optimization of the variable node degree distribution by fitting the EXIT curves of variable nodes and of the base code, for a target code rate. As in the previous section, let the degree distribution, renormalized over the degrees higher than 1, be denoted by $\tilde{\Lambda}(x)$,

$$\tilde{\Lambda}(x) = \sum_{i>1} \tilde{\lambda}_i x^{i-1} = \sum_{i>1} \frac{\lambda_i}{\sum_{j>1} \lambda_j} x^{i-1}.$$

Let $d_{cluster}$ be the average degree of clusters. Then, during the degree distribution optimization, the renormalized fraction $\tilde{\lambda}_2$ of edges connected to degree-2 variable nodes is required to be smaller than $2/d_{cluster}$, so that the average degree of G (which denotes the graph of codewords of partial weight 2) is smaller than 2. Suppose we have $\tilde{\lambda}_2 < 2/d_{cluster}$. Now we choose some structure on G (and, therefore, of the permutation of degree-2 variable nodes), so that G does not contain cycles. It seems that the simplest way would be to make G to be a union of disjoint paths. However, in this case the prediction of the iterative threshold given by the EXIT curve fitting is not accurate anymore and the reason is the following: the EXIT method implicitly assumes that the positions in C are chosen

to be of degree 2 independently of each other with probability $\tilde{\lambda}_2$. Therefore, the expected fraction of vertices of degree i in G should be $\binom{s}{i} \tilde{\lambda}_2^i (1 - \tilde{\lambda}_2)^{s-i}$ if all clusters are of size s . The prediction of the EXIT method is much more accurate if the degree 2 variables are chosen such that the fraction of clusters nodes of degree i is indeed this expected number. It remains to choose their positions in order to avoid cycles of sublinear length in the graph of codewords of partial weight 2. We provide a specific example of how this can be achieved in the next subsection.

B. Design of a Low-Rate Ensemble

The design criteria, proposed above, were previously used in the design of TLDPC codes of rates $1/3$ and $1/2$ [3], [4] and gave very good results. Iterative decoding thresholds of the ensembles are situated within $0.2 - 0.5$ dB from the Gaussian channel capacity. Moreover, it has been proved that one of the code ensembles has minimum distance growing linearly in the blocklength.

In this paper, we design a TLDPC ensemble of rate $1/10$, following the same construction methods. A particular advantage to use the TLDPC structure for the design of low-rate codes lies in the fact that it gives a faster decoding convergence. The reason is that one can allow a large non-zero fraction λ_1 and still satisfy the necessary condition on the linear minimum distance.

In what follows, a low-rate TLDPC base code and a permutation structure for degree-2 variable nodes are suggested.

1) *TLDPC base code of rate 1/2*: With the aim of designing codes of rates around $1/10$, we propose a TLDPC base code of rate $1/2$, defined by the Tanner graph shown in Figure 8.

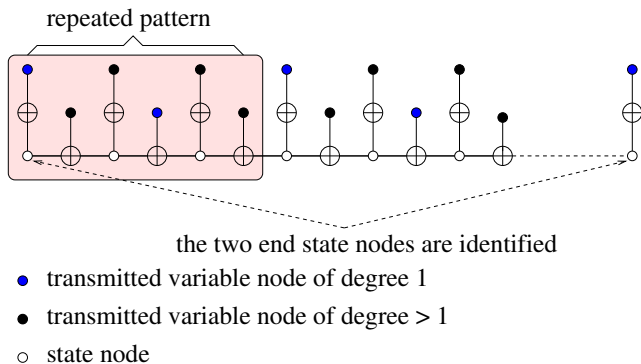


Fig. 8. Tanner graph of a TLDPC base code of rate $1/2$.

For the presented Tanner graph, $b_i = 1$ for any i . Moreover, each third section of the base code is chosen to be of degree 1, i.e. this position is connected to a degree-1 variable node in the bipartite graph. Positions of degree 1 are marked in blue in the figure. All other positions have degrees > 1 . Such a base code gives rise to a code ensemble with

$$\lambda_1 = \frac{1}{3}.$$

Let us consider how clusters look like in this case. It is easy to verify that the clusters correspond to the pattern in the Tanner graph of the base code represented in Fig. 9, as any two positions of degree > 1 in it give rise to a codeword of partial weight 2. The number of positions of degrees > 1 (i.e. the cluster degree) is 4. The corresponding graph of codewords of partial weight 2 contains as many clusters as there are such subgraphs in the Tanner graphs of the base code. To satisfy the necessary condition on the linear minimum distance given by Theorem 1, we should choose $\tilde{\lambda}_2$ such that:

$$\tilde{\lambda}_2 \leq \frac{1}{2}.$$

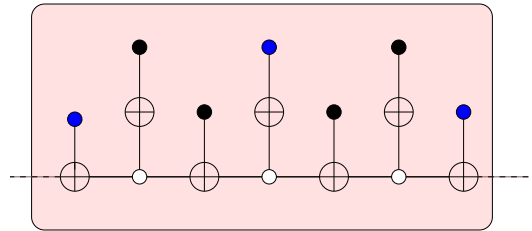


Fig. 9. Pattern in the Tanner graph of the TLDPC base code of rate $1/2$ giving rise to a cluster.

2) *Degree optimization over the Gaussian channel and permutation structure for rate 1/10*: Let us fix the design code rate equal to $1/10$. We choose $\tilde{\lambda}_2$ to be slightly less than $\frac{1}{2}$, namely $\tilde{\lambda}_2 = 0.4$ in order to simplify the structure of the graph of codewords of partial weight 2.

First we compute the cluster degree distribution $A = (a_0, a_1, a_2, a_3, a_4)$ where a_i represents the fraction of clusters of degree i in the graph of codewords of partial weight 2. If the degree of clusters in G are chosen at random given $\tilde{\lambda}_2 = 0.4$, the expected values of the a_i 's would be the following figures:

$$a_0 = \frac{81}{625}; \quad a_1 = \frac{216}{625}; \quad a_2 = \frac{216}{625}; \quad a_3 = \frac{96}{625}; \quad a_4 = \frac{16}{625}.$$

As explained before, we choose the a_i 's to be equal to these fractions in order to make the predictions given by the EXIT chart analysis more accurate.

We need to find a structure of G with this degree distribution, so that G does not contain cycles. We choose it to contain the following components which we call "stars", "twigs" and "chains" and which are shown in Fig. 10.

Divide the Tanner graph of the base code into subgraphs similar to the one represented in Fig.9 and associate a cluster to each of them. We assume that the number of clusters M is divisible by 625. The generation of the bipartite graph is then performed by associating clusters in order to form the aforementioned components. It is straightforward to check that this is indeed possible. We summarize in Table I the fraction of clusters consumed by each component. According to this table, there are only three points to check.

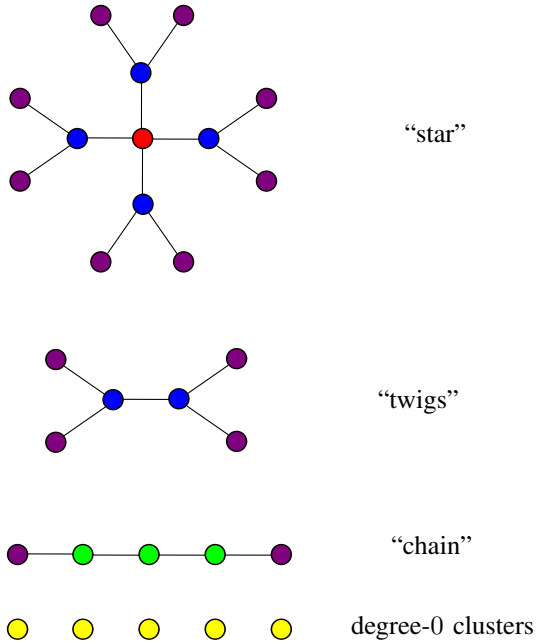


Fig. 10. Configurations in the structure of the graph of codewords of partial weight 2. Clusters of different degrees have a different color.

TABLE I

TABLE SHOWING HOW THE CLUSTERS ARE ARRANGED TO FORM THE COMPONENTS. AN ENTRY FOR A GIVEN COMPONENT c AND A GIVEN DEGREE i OF THE CLUSTER CORRESPONDS TO THE FRACTION OF CLUSTERS CONSUMED IN COMPONENT c WHICH ARE OF DEGREE i .

	0	1	2	3	4
“star”	0	$8a_4$	0	$4a_4$	a_4
“twig”	0	$2(a_3 - 4a_4)$	0	$a_3 - 4a_4$	0
“chain”	0	$a_1 - 2a_3$	a_2	0	0
“isolated cluster”	a_0	0	0	0	0

- 1) All clusters are consumed in the components (the sum of the entries of the column corresponding to degree i gives a_i).
- 2) Each entry should be nonnegative, this follows directly from the values taken by the a_i 's.
- 3) It is possible to form chains in this way. There is only one point to verify, namely that the number of clusters of degree 1 used to form chains is even. This is indeed the case since $(a_1 - 2a_3)M$ is clearly even.

After the degree optimization for the Gaussian channel performed with the EXIT chart method of [25], the following degree distribution was obtained:

$$\tilde{\Lambda}(x) = 0.4x + 0.264209x^2 + 0.090866x^4 + 0.236716x^8 + 0.008209x^9. \quad (3)$$

V. NUMERICAL RESULTS

As an example, we present performances of TLDPC codes of rate 1/10 and of lengths 6250, 18750 and 50000 over the Gaussian channel. In each of these cases, the degree

distribution (3) was adapted to the given blocklength. The corresponding word and bit error rates, obtained by simulations, are given in Fig.11. The maximum iteration number was fixed to 200.

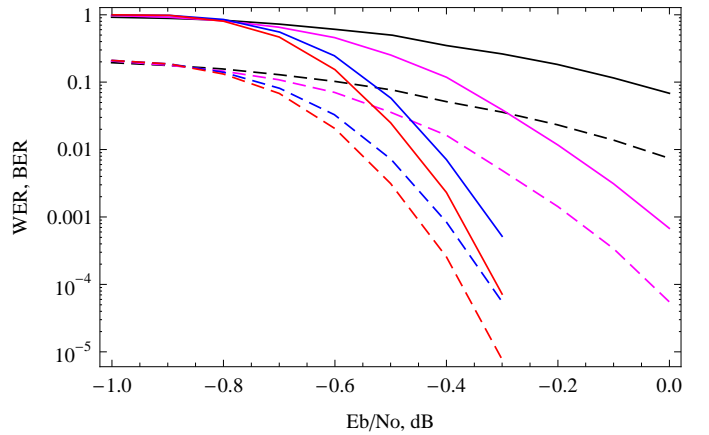


Fig. 11. Performance of TLDPC codes of rate 1/10 and of lengths (from right to left) 6250, 18750, 50000 and 62500 with $\lambda_1 = 1/3$ and the degree distribution (3). Solid lines represent word error rates and dashed lines - binary error rates.

It can be seen in the figure that the estimated decoding threshold is about -0.8 dB, which corresponds to the threshold obtained with the EXIT method. Notice that the threshold is only 0.5 dB away from channel capacity which is about -1.286 dB here. This is quite close for these signal to noise ratios, since the capacity at -0.8 dB is only about 0.111. In addition, numerical results did not catch the error-floor, which is expected to happen thanks to the good minimum distance of designed codes.

Concerning the decoding convergence, for the largest simulated blocklength (62500) and at signal-to-noise ratio -0.5 dB the decoder only needs 86 iterations in average in order to converge. Such a relatively fast convergence for these low-rates is due to the large fraction of degree-1 variable nodes in the code structure. Moreover, as the base code can be represented by a 2-state trellis, where each trellis section carries only one bit, the complexity of one decoding iteration is very low. This results in a low total decoding complexity of proposed TLDPC codes.

VI. DISCUSSION

In this paper we have followed two objectives. The first one was to define a necessary condition to design sparse-graph codes with linear minimum distance in the blocklength. Such a condition has been found and can be expressed either in terms of cycles or in terms of the average degree of the graph of codewords of partial weight 2.

The second objective was to design a new low-rate, structured code ensemble with such features as a linear minimum distance, a small gap to the channel capacity, a low decoding complexity and also a possibility to apply well-developed

techniques (EXIT charts, density evolution) to optimize the degree distribution of the variable nodes. The aforementioned design has been performed in the framework of TLDPC codes and a TLDPC code ensemble of rate 1/10 performing well over the Gaussian channel has been proposed.

The linear minimum distance property for the presented TLDPC ensemble may be proved by using standard techniques based on weight distributions, for instance by computing the growth rate of the average weight distribution in the asymptotic case and to show that its first derivative at the origin is strictly negative. We do not present such a proof in the paper, but we conjecture such a behavior.

VII. ACKNOWLEDGMENT

Part of this work was done when the first author was with France Telecom R&D.

REFERENCES

- [1] N. Alon, S. Hoory, and N. Linial. The Moore bound for irregular graphs. *Graphs Combin.*, 18:53–57, 2002.
- [2] A. Amraoui, A. Montanari, T. Richardson, and R. Urbanke. Finite-length scaling for iteratively decoded LDPC ensembles. *IEEE Trans. on Information Theory*, 55(2):497–473, February 2009.
- [3] I. Andriyanova, J.P. Tillich, and J.C. Carlach. Asymptotically good codes with high iterative decoding performances. In *ISIT'05*, pages 850–854. IEEE, September 2005.
- [4] I. Andriyanova, J.P. Tillich, and J.C. Carlach. A new family of asymptotically good codes with high iterative decoding performances. In *ICC'06*, June 2006.
- [5] A.Otmani and J.P. Tillich. On the minimum distance of generalized LDPC codes. in preparation, 2008.
- [6] A. Ashikhmin, G. Kramer, and S. ten Brink. Extrinsic information transfer functions : model and erasure channel properties. *IEEE Trans. on Information Theory*, 50(11):2657–2673, November 2004.
- [7] C. Berrou, A. Glavieux, and P. Thitimajshima. Near Shannon limit error-correcting coding and decoding. In *ICC'93*, pages 1064–1070, Genève, Switzerland, May 1993.
- [8] M. Breiling. A logarithmic upper bound on the minimum distance of turbo codes. *IEEE Transactions on Information Theory*, 50(8):1692–1710, 2004.
- [9] C. Di, T. Richardson, and R. Urbanke. Weight distribution of low-density parity-check codes. *IEEE Trans. Information Theory*, 52(11):4839–4855, November 2006.
- [10] C. Di, T. Richardson, and R. Urbanke. Weight distributions of Low-Density Parity-Check codes. *IEEE Transactions on Information Theory*, 52(11):4839–4855, November 2006.
- [11] D. Divsalar, S. Dolinar, and C. Jones. Low-rate LDPC codes with simple protograph structure. In *ISIT*, pages 1622–1626, Adelaide, Australia, 2005.
- [12] J. Ezri, A. Montanari, S. Oh, and R. Urbanke. The slope scaling parameter for general channels decoders and ensembles. In *Proc. of the IEEE Int. Symp. Information Theory*, pages 1443–1447, Toronto, Canada, July 2008.
- [13] J. Ezri, A. Montanari, and R. Urbanke. A generalization of the finite-length scaling approach beyond the BEC. In *Proc. of the IEEE Int. Symp. Information Theory*, pages 1011–1015, Nice, France, June 2007.
- [14] J. W. Lee. *The study of turbo codes and iterative decoding*. PhD thesis, Urbana, IL, 2003.
- [15] J. W. Lee and R. E. Blahut. Generalized EXIT chart and BER analysis of finite-length codes. In *Proc. IEEE Global Telecommunication Conf. (Globecom)*, pages 2067–2071, San Francisco, USA, December 2003.
- [16] J. W. Lee and R. E. Blahut. Convergence analysis and BER performance of finite-length turbo-codes. *IEEE Trans. on Communications*, 55(5):1033–1043, May 2007.
- [17] W. K. R. Leung, G. Yue, L. Ping, and X. Wang. Concatenated zigzag-Hadamard codes. *IEEE Trans. on Information Theory*, 52(4):1711–1723, April 2006.
- [18] A. Leverrier and P. Grangier. Unconditional security proof of long distance continuous-variable quantum key distribution. *Physical Review Letters*, 102(18):180504, 2009.
- [19] A. Otmani, J. P. Tillich, and I. Andriyanova. On the minimum distance of generalized LDPC codes. In *Proc. of the IEEE Int. Symp. Information Theory*, pages 751–755, Nice, France, June 2007.
- [20] H. Pishro-Nik and F. Fekri. Performance of low-density parity-check codes with linear minimum distance. *IEEE Trans. on Information Theory*, 52(1):292–300, January 2006.
- [21] T. Richardson and R. Urbanke. Multi-edge LDPC codes. Available at <http://lthcwww.ep.ch/papers/multiedge.ps>.
- [22] T. Richardson and R. Urbanke. Multi-edge LDPC codes. submitted to *IEEE Trans. on Inform. Theory*, 2005.
- [23] T. Richardson and R. Urbanke. *Modern coding theory*. Cambridge University Press, 2008.
- [24] A. Shokrollahi. New sequences of linear time erasure codes approaching the channel capacity. In *Proceedings of AAECC-13*, number 1719 in *Lecture Notes in Computer Science*, pages 65–76. Springer, 1999.
- [25] S. ten Brink. Convergence behaviour of iteratively decoded parallel concatenated code. *IEEE Trans. Commun.*, 49:1727–1737, Oct. 2001.
- [26] J. P. Tillich. The average weight distribution of Tanner code ensembles and a way to modify them to improve their weight distribution. In *Proceedings of ISIT'04*, page 7, Chicago, Illinois, 2004.
- [27] J.P. Tillich and G. Zémor. On the minimum distance of structured LDPC codes with two variable nodes of degree-2 per parity-check equation. In *Proceedings of ISIT 2006*, Seattle, USA, 2006.