Designing a Good Low-Rate Sparse-Graph Code

Iryna Andriyanova *Member, IEEE,* and Jean-Pierre Tillich *Member, IEEE*

***Abstract*—This paper deals with the design of low-rate sparse-graph codes, having a linear minimum distance d_{min} in the block-length n. Its main contributions are: a) a necessary condition on a general family of sparse-graph codes with linear d_{min}; b) a justification of having degree-1 bits in the low-rate code structure; c) a new, efficient ensemble of low-rate sparse-graph codes with bits of degree 1, designed so that the necessary condition (a) is satisfied.**

I. Introduction

Low rate codes play a crucial role in communication systems with very noisy transmission channels (e.g. power-limited sensor networks [2], ultra-wideband communication, code-spread Code-Division Multiple Access (CDMA) systems [3]) or in systems where an (almost) error-free communication is to be ensured (e.g. signalization channels, code-aided carrier frequency synchronization [4], etc.). Also, the use of low-rate codes can increase the operating range of a protocol, as it was demonstrated for continuous-variable quantum key distribution protocols in [5].

The most efficient practical coding schemes nowadays are based on sparse-graph codes, decoded using iterative decoding. However, the design of efficient low-rate sparse-graph codes is not so straightforward – it is well-known that standard Low-Density Parity-Check (LDPC) codes behave poorly at low rates –, and some special graph structures should be used for this. One of the first structured sparse-graph codes, proposed for low rates, is a rate-$1/10$ Multi-Edge (ME) LDPC ensemble from [6]. Its iterative decoding threshold is equal to -1.09 dB over the Gaussian channel, which is close to the theoretical limit (-1.286 dB for the rate $\frac{1}{10}$). Further, low-rate Accumulate-Repeat-Accumulate (ARA) LDPC codes were introduced in [7], with various code rates from $1/3$ to $1/10$. The ARA codes have good thresholds[1] and a simpler structure than ME-LDPC codes from [6]. However, their minimum distance d_{min} only grows as n^{α} with n being the codelength and $\alpha < 1$[2], while, for ME-LDPC codes, d_{min} is linear in n, which helps to lower the error probability after decoding in the so called error-floor region of channel parameters. For the completeness of picture, let us cite other low-rate sparse-graph code constructions: Serially-Concatenated Low-Density Generator Matrix (SC-LDGM) codes [9] of rate around $1/10$ (good threshold, but constant d_{min}); Turbo-Hadamard [10] and Zigzag-Hadamard [11] codes of rates from 0.05 to 0.00105 (good threshold, but complex structures with M Hadamard component codes and d_{min} of order $n^{(M-1)/M}$; various code families [10], [12], [13] for rates from $1/3$ to $1/12$ (good thresholds, and d_{min} at most logarithmic in n).

These examples illustrate that 1) it is difficult to design an efficient low-rate code with both good iterative threshold and minimum distance properties and 2) the existing codes, all obtained using different methods, do not provide a general approach for designing low-rate sparse-graph codes.

This paper aims at responding to both issues, by proposing general design conditions, that should be satisfied in order to have good error performance in both error-floor region (good channel conditions) and waterfall region (close to the iterative decoding threshold, bad channel conditions). The performance in the error-floor region being related to minimum distance properties, our aim here is to design code families with a linear minimum distance in the codelength (that is *asymptotically good code families*). For this purpose we first provide in Section II a simple necessary condition for a code family to be asymptotically good. On the other hand, the iterative decoding threshold and the waterfall performance in general can be improved by either having bits of degree 1 in the code structure (see [6], [11], [9]) or working over non-binary alphabets ([12], [13]). We follow the first approach, and propose to have a large fraction of degree-1 bits. Section III gives an insight on why degree-1 bits are of benefit and also how many of them should be taken in order to meet the previous necessary condition on d_{min}. Moreover, we use the proposed conditions and design a new class of low-rate sparse-graph codes, the so called Tail-Biting LDPC (TLDPC) codes with bits of degree 1, in Section V. The designed code ensemble is an extension of results, presented in [14], [1].

For the sake of simplicity, we focus ourselves on binary ensembles.

II. Necessary Condition for Linear Minimum Distance

In this section, a necessary condition for linear d_{min} is presented. It involves the average degree of a certain graph (what we call here the *graph of codewords of partial weight 2*), deduced from the Tanner graph of a sparse-graph code. The average degree should be less than 2 (see Theorem 1), otherwise d_{min} turns out to be at most logarithmic in n. Such a simple necessary condition seems to be a powerful tool for estimating d_{min}: it captures well the logarithmic behavior of d_{min} of parallel turbo-codes with two convolutional components (see Theorem 2), as well as all the known cases of asymptotically good LDPC codes (see [15]). Also note that, in the critical case when the average degree is exactly two,

I. Andriyanova is with the ETIS-UMR8051 group, ENSEA/University of Cergy-Pontoise/CNRS, 95015 Cergy, France, e-mail: iryna.andriyanova@ensea.fr.

J-P. Tillich is with Equipe-Projet SECRET, INRIA Roquencourt, France, e-mail: jeanpierre.tillich@inria.fr.

Parts of this paper were presented at the IEEE International Conference on Communications in 2006 [1].

[1]e.g. -1.03 dB over the Gaussian channel for the rate $1/10$

[2]$\alpha \approx 0.75$, see [8]

d_{min} is at most n^{α} with $\alpha < 1$, and that for certain structured LDPC code ensembles this represents the typical behavior of d_{min} (see [16]).

A. Common Representation for Sparse-Graph Codes

It will be convenient to bring in the following representation of sparse-graph codes, suggested in [17]. It covers many known code families such as LDPC codes, parallel turbo-codes, IRA or TLDPC codes.

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

(i) a binary code $\mathcal{B}$ of rate R_b of length m, with $m \geq n$. This code is called the *base code*;

(ii) a bipartite graph $\mathcal{G}$ between two sets V and W of vertices of size n and m respectively, where the degree of any vertex in W is 1 and the degree of the vertices in V is specified by a degree distribution $\Lambda = (\lambda_1, \lambda_2, \ldots, \lambda_s)$ where λ_i denotes the fraction of edges, incident to vertices of V of degree i.

Notice that each base code position is associated to exactly one variable node. A position in the base code $\mathcal{B}$ is said to have degree i if it is connected to a node of degree i in V. 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 assignments[3] of vertices of W belong to $\mathcal{B}$. The rate of the code obtained by this construction is given by the following lemma:

Lemma 2.1: The code rate is at least equal to the *design rate* R,

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

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

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

The proof of the lemma is given in Appendix A.

It is convenient to represent Λ by its polynomial form which is defined by $\Lambda(x) \overset{\text{def}}{=} \sum_{i=1}^{s} \lambda_i x^{i-1}$.

Most sparse-graph code constructions can be viewed as a particular instance of this construction:

Example: [LDPC codes] The LDPC base code is a juxtaposition of parity codes; $\Lambda(x)$ is the left degree distribution. $\diamond$

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

Example: [Parallel turbo codes] 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 subsets (information and redundancy). A node in V and a node in W can be connected only if they belong to the same subset type and all redundancy nodes have degree 1. $\diamond$

[3]a vertex in W receives the same assignment as the vertex in V it is connected to.

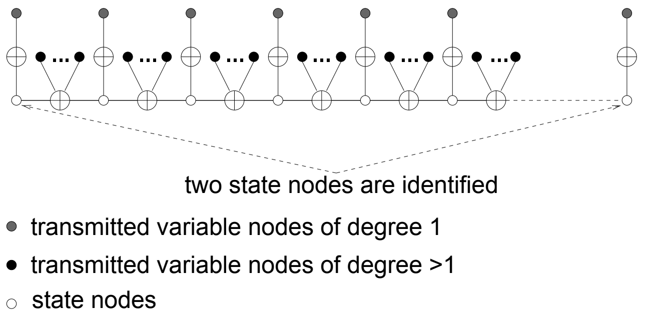
two state nodes are identified
● transmitted variable nodes of degree 1
● transmitted variable nodes of degree >1
○ state nodes

Fig. 1. Base code for systematic (I)RA codes and standartized LDPC codes.

The iterative decoding procedure [18] for sparse-graph codes is the following. At each iteration, base code decoding is performed in order to get extrinsic messages for bits of $\mathcal{B}$, then intrinsic messages at the variable node side are calculated. After some number of iterations, a posteriori messages of code bits are computed. The decoding complexity therefore depends on 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 (i.e. speed of decoding convergence).

Remark 1: The same sparse-graph code may have different representations in terms of $\mathcal{B}$ and $\mathcal{G}$, which will result in a different decoding schedule and, thus, in different decoding performance. For example, Repeat-Accumulate (RA) codes [19] can be decoded either as LDPC codes or as turbo codes. RA codes, viewed as LDPC codes, have a base code which is a juxtaposition of parity nodes, exactly as in the LDPC case. RA, viewed as turbo codes, have a base code with a Tanner graph presented in Fig.1, and the variable nodes of degree 2, connecting parity $\oplus$ nodes, become state nodes in the base code and correspond to variable nodes of degree 1 in the bipartite graph. Note that in the first case, there are no variable nodes of degree 1 in the bipartite graph.

B. Graph of Codewords of Partial Weight 2

Note that here we allow some variable nodes to be of degree 1. Therefore, we allow positions of degree 1 in $\mathcal{B}$. The location of these positions has a crucial impact on the minimum distance of the overall code, which may become constant in the worst case. In what follows, this case is supposed to be avoided by making the following assumption

Assumption 1: There are no codewords in $\mathcal{B}$, whose support is only formed by degree-1 positions.

To study d_{min}, let us start with some definitions.

Definition 2 (Codewords of $\mathcal{B}$ of partial weight 2): Codewords of $\mathcal{B}$ of partial weight 2 are the codewords, which have 1's at exactly two positions of degree > 1.

In other words, the support of a codeword of partial weight 2 contains some number of positions of degree 1 and only two positions of degree > 1.

Example: [Codewords of $\mathcal{B}$ of partial weight 2 for LDPC and RA codes] For classical LDPC codes, having $\lambda_1 = 0$, any codeword of $\mathcal{B}$ of weight 2 is a codeword of partial weight 2. Note that such a codeword correspond to two bits of the same

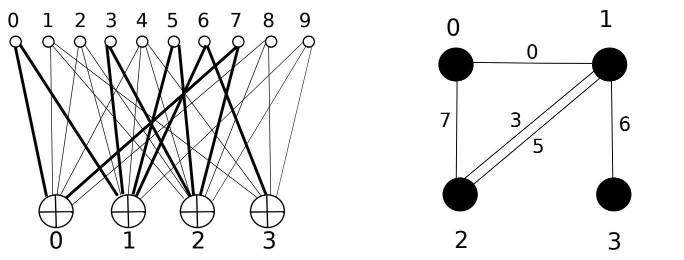

Fig. 2. Bipartite graph $\mathcal{G}$ of an LDPC code (left) and its corresponding graph G (right). Edges of $\mathcal{G}$, represented by thick lines, are connected to variable nodes of degree 2.

parity code set to 1 (so that the parity equation is satisfied) and all others to 0. For the base code of RA codes (Fig.1), all codewords of $\mathcal{B}$ of partial weight 2 are obtained by the following procedure. Pick any two positions i and j in $\mathcal{B}$ of degree > 1 and set them to 1. Also, set to 1 all the positions of degree 1, that correspond to state nodes between the parity checks containing i and j. Let all other positions be set to 0. If the number of chosen state nodes is k, then we have constructed a codeword of BC of weight $2 + k$, which is a codeword of partial weight 2. $\diamond$

A position of degree > 1 in $\mathcal{B}$ can be contained in the support of several codewords of partial weight 2 (e.g. a fixed position of a parity code of length d is contained in $d - 1$ supports of codewords of weight 2). The equivalence classes of those positions form *clusters*:

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

Example: For classical LDPC 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. For RA codes, it is easy to check in Fig.1, that any two positions of degree > 1 form the support of a codeword of partial weight 2. Hence, the base code of a RA code corresponds to one big cluster, and it contains all positions of degree > 1. $\diamond$

Now we are ready to define the *graph of codewords of partial weight 2*, which will serve us as the main object in the necessary condition on d_{min}:

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, where $\tilde{V}$ is 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 k and l of $\mathcal{B}$, belonging to the clusters $\tilde{v}_i$ and $\tilde{v}_j$ respectively, which are connected to the same variable node of degree 2 in the bipartite graph $\mathcal{G}$.

As we can see from the definition, G is constructed using clusters in $\mathcal{B}$ and variable nodes of degree 2 in $\mathcal{G}$.

Example: [Graph G for LDPC and RA codes] G for classical LDPC codes is given in Fig. 2. Vertices of G (i.e. clusters) correspond to parity nodes in the Tanner graph of the LDPC code. Two nodes in G are connected if their corresponding parity nodes are connected through a degree-2 variable node in $\mathcal{G}$. For RA codes with variable node degrees > 2, the graph G has only one node, because there is one single cluster. $\diamond$

C. Cycles and Average Degree of G

It is well known [15] that the first source of low weight codewords in an unstructured LDPC code are cycles in its Tanner graphs, only containing variable nodes of degree 2. Let us show that they correspond to cycles in G. Once it is done, the necessary condition on linear d_{min} will follow easily. We start with the definitions of node and cycle weights:

Definition 5 (Node weight $w^v_{i,j}$ and codeword $\mathbf{c}^v_{i,j}$): For a node v in $\tilde{V}$ and two edges i and j connected to it, define a node weight $w^v_{i,j}$ as follows. By definitions of cluster and of G, these two edges correspond to two positions of degree 2 and they are contained in the support of a codeword of $\mathcal{B}$ of partial weight 2 which involves a certain number a of positions of degree 1. Let $w^v_{i,j}$ be equal to this number a and define $\mathbf{c}^v_{i,j}$ to be this codeword of partial weight 2. Notice that we have $|\mathbf{c}^v_{i,j}| = 2 + w^v_{i,j}$.

Definition 6 (Cycle weight): The weight of a cycle $\mathcal{C} = (V_\mathcal{C}, E_\mathcal{C})$ in G is equal to $|E_\mathcal{C}| + \sum_{v \in V_\mathcal{C}} w^v$, where w^v is the node weight associated with vertex v in $V_\mathcal{C}$ and the two edges in $E_\mathcal{C}$ connected to v.

Example: Classical LDPC codes give a trivial example of node weights as $w^v_{i,j} = 0$ for any node v and positions i and j in G, because $\lambda_1 = 0$. Hence, the cycle weight of a cycle $\mathcal{C}$ is simply $|E_\mathcal{C}|$, i.e. the number of degree-2 variable nodes in the corresponding cycle in $\mathcal{G}$. Thus, in Fig.2 for instance, the cycle between clusters 0, 1 and 3 has weight 3 and the cycle between clusters 1 and 2 has weight 2.

As an example of $\lambda_1 > 0$, let us consider the same graph G in Fig.2, but assuming that, say, $w^0_{0,7} = w^1_{0,3} = w^2_{3,7} = 1$, $w^1_{3,5} = 2$ and $w^2_{3,5} = 3$. Then the cycle between clusters 0, 1 and 2 has weight $3 + 3 = 6$ and the cycle between clusters 1 and 2 has weight 7. $\diamond$

Here is the fundamental relation between cycles in G and low-weight codewords of $\mathcal{B}$:

Proposition 1: A cycle of weight l in G induces a codeword of weight at most l in the sparse-graph code.

Proof: If $\mathcal{C} = (V_\mathcal{C}, E_\mathcal{C})$ is a cycle in G, we associate to it a codeword $\mathbf{c}$ of $\mathcal{B}$, obtained as the sim of the $\mathbf{c}^v_{i,j}$'s over all vertices v of the cycle (where we denote by i and j the two edges adjacent to v). The weight of $\mathbf{c}^v_{i,j}$ is equal to $2 + w^v_{i,j}$ by definition of node weights, so the weight of $\mathbf{c}$ is upper-bounded by $2|E_\mathcal{C}| + \sum_{v \in V_\mathcal{C}} w^v$ (the number of the degree 2 base code positions involved in it is exactly $2|E_\mathcal{C}|$ while the number of the degree 1 base code positions is upper-bounded by $\sum_{v \in V_\mathcal{C}} w^v$). Note that the degree-2 variable nodes, involved in $\mathbf{c}$, are counted twice, while the degree-1 variable nodes are counted only once. In other words, $\mathbf{c}$ yields a codeword of the sparse graph code of weight at most $|E_\mathcal{C}| + \sum_{v \in V_\mathcal{C}} w^v$. ■

The weight of the smallest cycle in G is an upper bound on d_{min} of the sparse graph code. Therefore:

Corollary 1: If all the node weights $w^v_{i,j}$ of a given graph G are smaller than some constant a, then the minimum distance of its corresponding sparse-graph code is upper bounded by $(a + 1)|E_\mathcal{C}|$.

From this corollary we deduce one of our main results:

Theorem 1 (Upper bound on d_{min}): Consider a sparse-graph code ensemble of length n for which the corresponding graphs of codewords of partial weight 2 have node weights $\leq a$, where a is some constant. If the average degrees of these graphs are greater than $2+\epsilon$ for some $\epsilon > 0$, then the minimum distance of a code from the ensemble is $O(\log n)$.

Proof. Consider a sparse graph code. 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 [20], saying that the number of vertices n in G satisfies $n \geq 2\frac{(\Delta-1)^t-1}{\Delta-2}$, with $t = \lfloor \frac{g}{2} \rfloor$. Hence, $t \leq \log_{\Delta-1}\left(\frac{\Delta-2}{2}n+1\right)$. We conclude that $d_{\min} \leq (a+1)g \leq (a+1)(2t+1) \leq (a+1)\left(2\log_{\Delta-1}\left(\frac{\Delta-2}{2}n+1\right)+1\right)$. □

D. Necessary Condition

The following necessary condition follows immediately from Corollary 1 and Theorem 1:

To avoid a logarithmic minimum distance, the average degree Δ of the graph of codewords of partial weight 2 should satisfy $\Delta \leq 2$.

Example: Consider LDPC codes. Let $\hat{\lambda}_2$ be the fraction of its degree-2 variable nodes and let $\bar{\rho}$ be the average degree of its check nodes ($\bar{\rho} = \frac{m}{r}$, where r is the number of check nodes and m is the number of edges). The number of clusters is equal to r. To satisfy the necessary condition, G should not have more than r edges. So, there should be at most r variable nodes of degree 2 in the bipartite graph. As there are $\frac{\hat{\lambda}_2 m}{2}$ of degree-2 nodes in total,

$$\frac{\hat{\lambda}_2 m}{2} \leq r = \frac{m}{\bar{\rho}}, \quad \text{and} \quad \hat{\lambda}_2\bar{\rho} \leq 2.$$

Therefore if we want a linear d_{min}, one should guarantee $\hat{\lambda}_2\bar{\rho} \leq 2$. $\diamond$

By now, only codes with bounded node weights were considered. However, our results can be extended to codes with unbounded weights, and Corollary 1 and Theorem 1 will still hold. For completeness of demonstration, let us elaborate a bound for parallel turbo codes, which leads to a much shorter proof of the result by Breiling [21]:

Theorem 2 ([21]): The minimum distance of parallel turbo codes with two components grows at most logarithmically in the blocklength n.

Proof: For simplicity, assume equal component codes, and that they are recursive, systematic, convolutional and of type $(n,1)$. Then, there exists $t > 0$, 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 $2t$ clusters (the factor 2 shows that there are two convolutional codes and, therefore, two sets of clusters), they are connected through N edges, N being the number of information bits in the turbo code.

Note that the node weights of 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' of $2N/3+O(1)$ clusters and of degree 3. Node weights of G' are bounded by $2w$. So, G' has a cycle of length at most $2\log_2(2N/3+O(1))$ and of weight at most $2(1+2w)\log_2(2N/3+O(1))$. By Proposition 1, this yields a codeword of weight $2(1+2w)\log_2(2N/3+O(1))$. ■

III. On Having Bits of Degree 1

After establishing a necessary condition on d_{min} to deal with error-floors, let us now consider how it is possible to improve the performance in the waterfall region. Let us quote [18]: *"Given the importance of degree-two edges, it is natural to conjecture that degree-one edges could bring further benefits"*. Indeed, this statement goes well with the observation that turbo codes have in general a better decoding convergence than LDPC codes, and tend to outperform them at short lengths. Note that turbo codes have bits of degree 1, but not LDPC codes. Another illustration is provided by [6, Table VIII]: where an LDPC code with a small fraction of bits of degree 1 has a better (i.e. steeper) waterfall performance than conventional LDPC codes without degree-1 bits.

The issue of having a good decoding convergence is even more present at low code rates: the number of decoding iterations is inversely proportional to the code rate R, and their number may go up to several hundreds (!). So, in order to design a good low-rate ensemble, it seems to be necessary to have bits of degree 1[4]. To show that this is indeed true, let us investigate what happens with the number of decoding iterations and the steepness of the error performance curve in the waterfall region when a fraction λ_1 of bits of degree 1 is non-zero.

Let us give a heuristic explanation with the help of an EXIT chart over the binary erasure channel[5] with erasure probability p (BEC(p)). Over the BEC, the EXIT chart predicts accurately the infinite-length behavior of the code ensemble. It also represents the "average" trajectory for finite blocklengths, consisting of horizontal and vertical steps between two EXIT curves, for variable nodes and for the base code respectively (see for example Fig. 1 in [22]). Note that the EXIT curve of variable nodes always depends on p, and the EXIT curve of the base code may or may not depend on p. Given two EXIT curves for some erasure probability p, the iterative decoding typically succeeds[6] iff the curve of variable nodes is above of the curve of the base code. The channel erasure probability p^*, for which two EXIT curves touch each other, is called the *asymptotic iterative threshold*. Moreover, the area $\Delta\mathcal{A}(p)$ between both EXIT curves for erasure probability p has two

[4]or "hidden" bits of degree 1 in the case of LDPC codes decoded in a turbo-like manner, namely LDPC codes for which all parity-checks involve at least two bits of degree 2.

[5]Note that the same kind of explanation can also be given for other binary memoryless symmetric (BMS) channels by asserting that the fundamental relation, namely Theorem 3, which holds for the binary erasure channel, holds approximately for BMS channels.

[6]i.e. succeeds with probability tending to 1 as $n \to \infty$

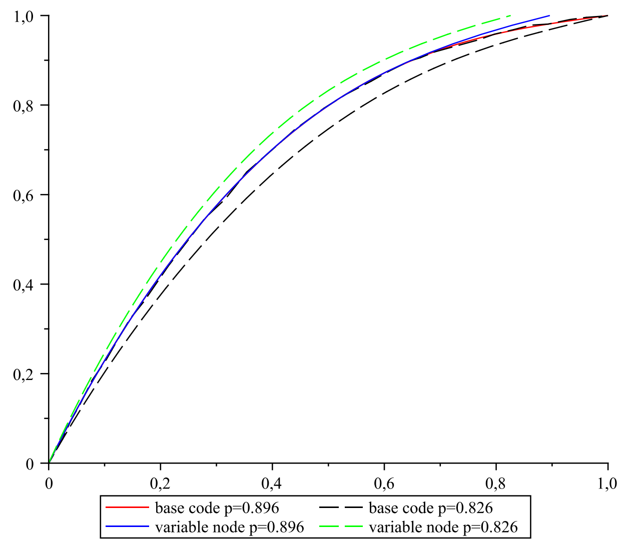

Fig. 3. EXIT chart of a TLDPC code of $R = \frac{1}{10}$ code with $\lambda_1 = \frac{1}{3}$. The threshold for iterative decoding is about $p \approx 0.896$ and the EXIT curve of the variable nodes almost coincides at the threshold with the EXIT curve of the base code (a difference can only be observed at the rightmost part where the variable node curve is slightly above the base code curve). On the other hand, the top curve corresponds to the EXIT curve of the variable nodes for $p = 0.826$ and is quite distinct from the bottom curve which corresponds to the EXIT curve of the base code at $p = 0.826$. Notice that unlike LDPC codes where the base code curve corresponds to a single-parity check code and does not depend on the probability of erasure of the channel, this is not the case here : when p decreases, the base code curve moves away from the base code curve at the threshold.

nice properties : a) for $p = p^*$, $\Delta\mathcal{A}(p^*)$ is proportional to the gap to capacity; b) for $p < p^*$, $\Delta\mathcal{A}(p)$ is related to the speed of decoding convergence.

Let us see examine now $\Delta\mathcal{A}(p)$ depends on λ_1. We start by defining the EXIT curves of variable nodes and of the base code. For convenience, the definitions will be given in terms of the entropy, following [18], and not in terms of mutual information, as in [22], even though both representations are equivalent.

1) *EXIT curve of variable nodes:* for $\lambda_1 = 0$, it is given by the set of points $(p\lambda(x), x)$, $x \in [0, 1]$. For $\lambda_1 > 0$, it is given by the set of points $\left\{(\frac{p(\lambda(x)-\lambda_1)}{1-\lambda_1}, x), x \in [0, 1]\right\}$. In a more compact notation, if one defines renormalized fractions of edges of degrees > 1 as

$$\tilde{\lambda}_i \overset{\text{def}}{=} \begin{cases} 0, & i = 1; \\ \frac{\lambda_i}{1-\lambda_1}, & i > 1; \end{cases} \quad (1)$$

and the associated polynomial $\tilde{\Lambda}(x)$ as

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

then the EXIT chart of variable nodes is given by $\left\{(p\tilde{\Lambda}(x), x), x \in [0, 1]\right\}$.
As an example, two EXIT charts of variable nodes for $p = 0.826$ and $p = 0.896$ of a TLDPC ensemble are given in Fig. 3.

2) *EXIT chart of the base code:* it is defined as the curve, relating the fraction of erased messages, ingoing to the base code, to the fraction of outgoing erased messages after the base code decoding, when $n \to \infty$. In some cases, this EXIT curve can be described analytically – for example, for a right-regular LDPC code, it is given by the set of points $\left\{(x, 1 - (1 - x)^{r-1}), x \in [0, 1]\right\}$. Notice that the EXIT curve of the base code depends on p only if $\lambda_1 > 0$. Otherwise it does not change the form with p. For an example, two EXIT charts of the base code of a TLDPC ensemble with $\lambda_1 > 0$ are given in Fig. 3 for $p = 0.826$ and $p = 0.896$ respectively.

The theorem below follows directly from the results of [22], even though it was not explicitly stated there:

Theorem 3: [Area theorem] Let $\Delta\mathcal{A}(p)$ be the area between the two EXIT curves. Then

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

where $C(p)$ is the capacity for BEC(p), $C(p) = 1 - p$. The proof of Theorem 3 is given in Appendix.

It follows immediately that:

Corollary 2:

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

This result implies the following:

- As observed in [22], for a capacity-achieving sequence of codes in the sense of [23], the quantity $\Delta\mathcal{A}(p) \to 0$ as $p \to p^*$. Notice that for some fixed gap to capacity and $\bar{\lambda}$, the area between two EXIT curves of variable nodes and of the base code is proportional to $\frac{1}{1-\lambda_1}$. Hence, $\Delta\mathcal{A}(p^*)$ for codes with $\lambda_1 > 0$ is smaller than for codes with $\lambda_1 = 0$, meaning that the iterative threshold of codes with $\lambda_1 > 0$ lies closer to the capacity.
- Although the number of iterations does not necessarily decreases with $\Delta\mathcal{A}(p)$ (it may also depend on the shape of both EXIT curves), in many cases it actually does. The presence of degree-1 nodes makes two EXIT curves lie far from each other for any $p > p^*$, which decreases the number of iterations and thus improves the decoding convergence. Note that turbo-codes, especially low rate turbo-codes, have a large λ_1. This may explain the small amount of iterations needed for their convergence, in comparison to LDPC codes, decoded by the standard Gallager algorithm and not having degree-1 bits at all.
- A fast increase of $\Delta\mathcal{A}(p)$ with p increases the slope of the error curve in the waterfall region, as it was put forward in [24], [25], [26]. This might be the explanation why turbo-codes are believed to outperform LDPC codes for moderate codelengths: they have a steeper waterfall performance[7].

As an illustration, let us consider a particular TLDPC ensemble, which will be defined a bit later in Section V. It

[7]We refer the reader to [27] for a rigorous derivation of the exponential behavior of the error probability, shown on the example LDPC codes over the BEC. The generalization of the result on turbo-like ensembles is given in [28]. An extension of [27] to general channels is given in [29], [30].

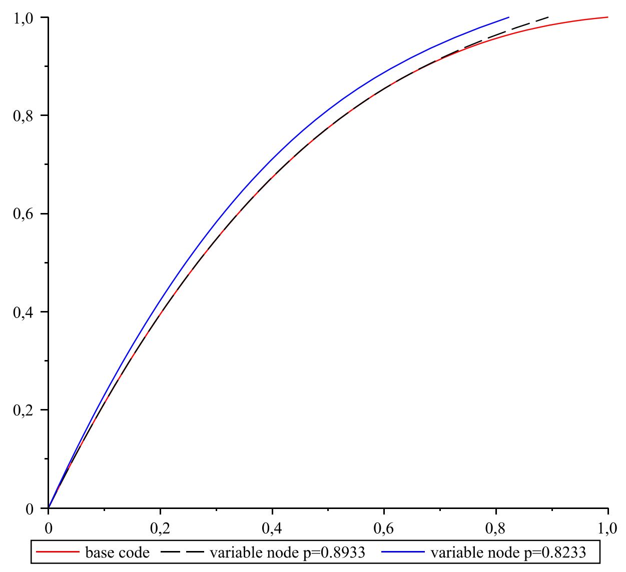

Fig. 4. EXIT chart of an LDPC code of rate $\frac{1}{10}$ code.

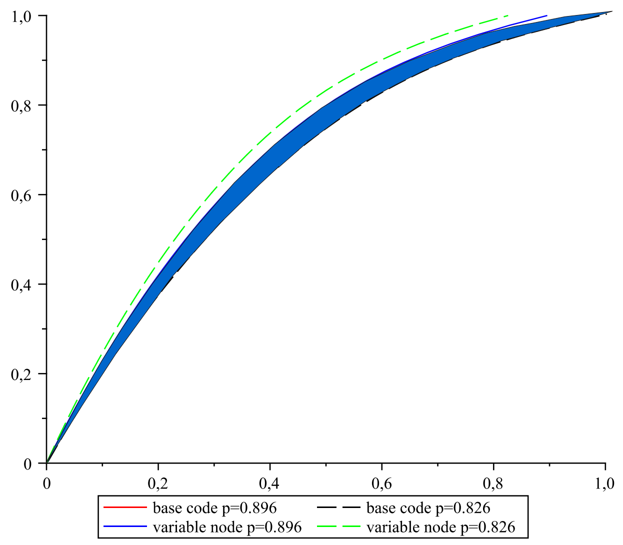

Fig. 5. $\Delta\mathcal{A}_1$ for the TLDPC code of rate $\frac{1}{10}$.

is an ensemble of rate $R = \frac{1}{10}$, with $\lambda_1 = \frac{1}{3}$. The ensemble is almost capacity-achieving for the BEC: it corrects up to 89.6% channel erasures. Its two EXIT curves for $p_0 = 0.896$ are drawn by solid lines in Fig.3 and, indeed, they touch each other. Fig.3 also presents two EXIT curves (dashed lines) for $p = p_0 - 0.07 = 0.889$. One can see that they moved from each other, as predicted. To estimate the speed of moving of the EXIT curves, let us compare them with the EXIT curves of an LDPC code ensemble of rate $\frac{1}{10}$. Let it be an LDPC ensemble with check nodes of degrees 2, 3 and 4, the edge connections to which are described by the check degree distribution $\rho(x) = \frac{1}{10}x + \frac{1}{2}x^2 + \frac{2}{5}x^3$ (see [18] for definition of $\rho(x)$). Such a choice of $\rho(x)$ makes the shapes of EXIT curves for the TLDPC base code and for the LDPC base code similar to each other, for a fair comparison. To design an LDPC code with parameters similar to those of the TLDPC code ($R \approx \frac{1}{10}$ and maximum variable node degree is 12), let $\Lambda(x)$ be $0.486x + 0.165x^2 + 0.037x^3 + 0.15x^4 + 0.132x^{10} + 0.03x^{11}$. The ensemble has iterative threshold $p_0 \approx 0.8933$.

Fig.4 shows two EXIT curves of the LDPC ensemble at p_0 and for $p = p_0 - 0.07$. At $p = p_0 - 0.07$, the EXIT curves of the base code and of variable nodes are much closer than they are in the TLDPC case, because the EXIT curve of the base code does not change with p: it is always given by the function $x \mapsto 1 - \rho(1-x)$.

The situation becomes different when $\lambda_1 > 0$ (the TLDPC case). When the channel improves, the EXIT curve of the base code moves below. The gain in the area is quantified by Proposition 2, and the area $\Delta\mathcal{A}_1$ between the EXIT charts of the base code at p_0 and at $p_0 - \Delta p$ is given by

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

Fig.5 shows this area difference. Thereforem there is a big difference between the TLDPC and the LDPC cases, and a TLDPC code clearly would need a smaller number of decoding iterations. Moreover, as the EXIT curves of the base code and of variable nodes lie further apart for TLDPC codes, they are very likely to have a steeper waterfall performance.

Although the expression from Proposition 1 seems to depend on λ_1 too, this quantity has *no* influence on how fast the variable node curve moves away when p decreases. Indeed, the area $\Delta\mathcal{A}_2$ between the EXIT curves of variable nodes at p_0 and at $p_0 - \Delta p$ is given by

$$\Delta\mathcal{A}_2 = \frac{\frac{1}{\tilde{\tilde{\lambda}}} - \lambda_1}{1-\lambda_1}\Delta p = \frac{\Delta p}{\tilde{\tilde{\lambda}}},$$

where $\tilde{\tilde{\lambda}} \overset{\text{def}}{=} \left(\sum_{i>1}\frac{\tilde{\lambda}_i}{i}\right)^{-1}$ and $\tilde{\lambda}_i$'s are given by (1). This is a consequence of the fact that the EXIT chart of variable nodes actually depends on $\tilde{\Lambda}(x)$, given by (2), and not on $\Lambda(x)$. The dependency on $\tilde{\tilde{\lambda}}$ suggests that, in order to improve the iterative eeror performance, one should have $\tilde{\tilde{\lambda}}$ as small as possible. Ideally, one would get $\tilde{\tilde{\lambda}} = 2$, which is, by the way, the case of parallel turbo-codes. This consideration provides a heuristic explanation of the common belief that sparse graph codes with small $\tilde{\tilde{\lambda}}$ provide a good waterfall performance at small and moderate code lengths. Also, $\tilde{\tilde{\lambda}} = 2$ implies $\tilde{\Lambda}(x) = x$. Hence, the EXIT curve of variable nodes becomes the straight line $x = py$. So, a (almost) capacity-achieving ensemble for this case should have a base code, the EXIT curve of which is close to $x = py$. This behaviour is possible to obtain, and this is how we designed the TLDPC code ensemble, that we define in the next section.

IV. Our Approach to Design Low Rate Codes

Combining ideas from two previous sections, one gets two design conditions for a low-rate, sparse-graph code ensembles: a) one should have $\lambda_1 > 0$ and a large fraction of λ_2, and b) the corresponding graph G should not have cycles. These two conditions are contradictory, and some tradeoff should be found. The contradiction comes from the fact that increasing λ_1 implies increasing the size of clusters in G, which in its turn increases the average degree Δ, and decreases the growth of d_{min} (see Theorem 1). Therefore, a code ensemble with small λ_1 and λ_2 has a good chance to have a good d_{min} but a bad iterative threshold. From another hand, a code ensemble with large λ_1 will have fast decoding convergence and potentially a good iterative threshold[8], but bad minimum distance properties.

We try to achieve a tradeoff with the following procedure:

1) Fix a fraction λ_1.
2) Choose a base code $\mathcal{B}$ of rate R_b so that it has the fraction λ_1 of positions of degree 1 and the following is verified:
   (a) the clusters, formed by codewords of partial weight 2 in $\mathcal{B}$, have bounded weights,
   (b) the EXIT chart of the base code is close to the straight line.

   Note that minimizing the size of clusters maximizes the maximum fraction of edges of degree 2 (denote it by λ_2^{max}), while the necessary condition on linear d_{min} growth is still satisfied. The constraint (b) allows us to minimize $\Delta\mathcal{A}(p^*)$ and thus to improve the iterative threshold.
3) Given $\mathcal{B}$, optimize the degree distribution $\Lambda(x)$ in order to maximize the design rate[9] R, under condition $\lambda_2 < \lambda_2^{max}$. Instead of considering $\lambda(x)$, one can equivalently consider $\tilde{\Lambda}(x)$, under the equivalent condition $\tilde{\lambda}_2 < \tilde{\lambda}_2^{max}$.
4) Define the structure for the permutation of edges, connected to degree-2 variable nodes in the bipartite graph $\mathcal{G}$, so that
   (a) the corresponding graph G does not contain any cycle,
   (b) if all clusters are of size s, then the fraction of clusters of degree i is $a_i = \binom{s}{i}\tilde{\lambda}_2^i(1-\tilde{\lambda}_2)^{s-i}$.

   Here, the constraint (a) comes from the necessary condition on d_{min}. The constraint (b) is needed to keep the prediction of the iterative decoding threshold accurate: the optimization of $\lambda(x)$ by EXIT charts or density evolution implicitly assumes that the positions of degree 2 in $\mathcal{B}$ are chosen independently of each other with probability $\tilde{\lambda}_2$. So, the expected fraction of clusters of size i in G should be equal to a_i.
5) Randomly generate the permutation for edges connected to variable nodes of degrees > 2.

The design procedure above is aimed to satisfy conditions of Theorems 1 and 3, and also to allow the one-dimensional optimization of $\Lambda(x)$, even in the presence of a structured permutation (condition 4(b)). It was previously used for the design of TLDPC codes of rates $1/3$ and $1/2$ in [14], [1], and already gave very good results. Moreover, it has been proved that one of the code ensembles from [14] has linear d_{min}.

[8]under condition that the base code has been appropriately chosen

[9]using EXIT charts or density evolution

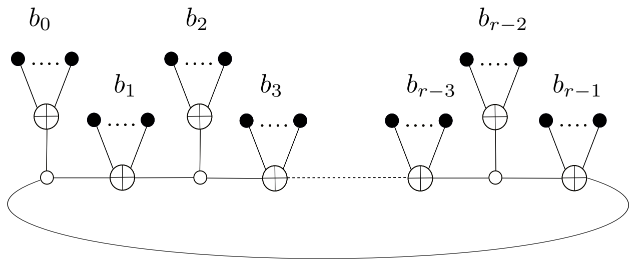

Fig. 6. Tanner graph of a TLDPC base code.

Let us design a TLDPC ensemble of low rate ($R = 1/10$), following the same design procedure.

V. TLDPC Ensemble of Rate $1/10$ Satisfying the Necessary Condition on d_{min}

TLDPC codes are structured codes, first proposed in [14] to meet the requirements of a low iterative decoding complexity, of good d_{min} and of small gap to the channel capacity. They can be viewed as a slight modification of LDPC codes, allowing to have degree-1 variable nodes by adding some state nodes to the graph structure. They differ from the multi-edge approach suggested in [6] by: (i) the TLDPC base code is not a juxtaposition of parity-check codes but a tail-biting convolutional code with binary state nodes, (ii) its permutation structure allows a one-dimensional optimization of $\tilde{\Lambda}(x)$, and not a multi-dimensional optimization as for ME-LDPC codes.

1) Definition: For the moment, suppose that $\lambda_1 = 0$. Then the TLDPC base code is defined as follows:

Definition 7 (TLDPC base code): The base code $\mathcal{B}$ of the TLDPC code is a tail-biting convolutional code, the Tanner graph of which is presented in Fig.6. $\bullet$'s are associated with positions in $\mathcal{B}$, white vertices with non-transmitted states, and $\oplus$'s represent parity-check equations. The first and the last state nodes are identified. The number of $\bullet$'s associated to the i-th $\oplus$ is denoted by b_i.

For $\lambda_1 > 0$, the TLDPC base code is defined in a similar matter, yet the positions of degree 1 in $\mathcal{B}$ have to be specified. It is interesting to mention that systematic RA codes, systematic IRA codes (Irregular Repeat-Accumulate) codes and most of the LDPC codes, which are standardized[10] are in fact a subclass of TLDPC codes, once they are decoded as turbo-coded and not as LDPC codes. All these codes have particular TLDPC base codes (see Fig. 1), for which all b_i's are equal to 1 for even values of i 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.

The important feature of the EXIT curve for the defined TLDPC base code is that it is close to a straight line. Moreover, the base code is not more complex to decode than single parity-check codes, and is much easier to decode than convolutional codes within the turbo structure. Compared with conventional LDPC codes, the TLDPC base code allows to have a larger λ_2 under the necessary condition on d_{min}.

[10]i.e. those LDPC codes which have the same amount of degree-2 variable nodes as there are of parity-check nodes and where the parity-check nodes are connected together via a single chain of degree 2 variable nodes.

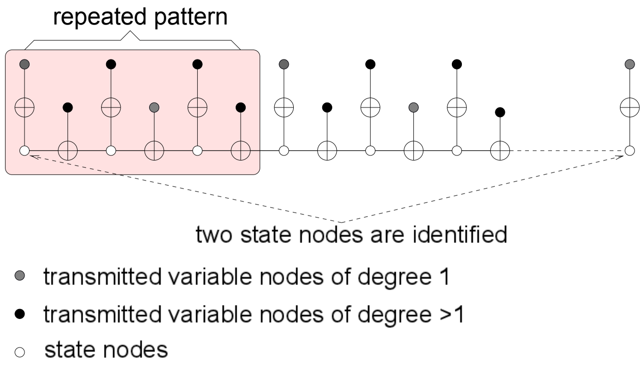
repeated pattern
two state nodes are identified
● transmitted variable nodes of degree 1
● transmitted variable nodes of degree >1
○ state nodes

Fig. 7. Tanner graph of a TLDPC base code of rate $1/2$.

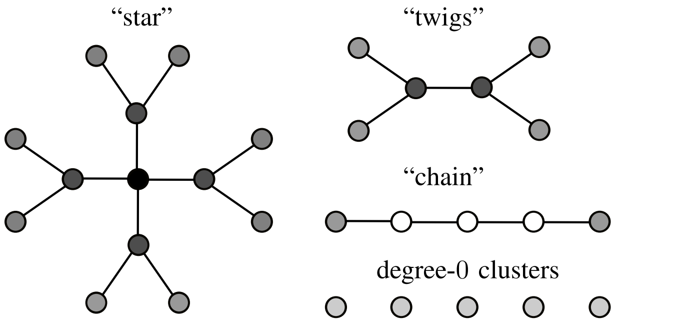

Fig. 8. Configurations in the structure of the graph of codewords of partial weight 2. Clusters of different degrees have a different color.

A. Design of a Low-Rate Ensemble

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 low rates, we propose a TLDPC base code of rate $1/2$, defined by the Tanner graph from Fig.7. Here $b_i = 1$ for any i. Each third section of the base code is chosen to be of degree 1, i.e. these positions are connected to degree-1 variable nodes in $\mathcal{G}$. All other positions in $\mathcal{B}$ have degrees > 1. Therefore, by choosing such a base code, this fixes λ_1 to $\frac{1}{3}$.

As for the clusters in the graph G, they correspond to a pattern in the Tanner graph of $\mathcal{B}$, shown in Fig. 9; one can check that any two positions of degree > 1 within the pattern give rise to a codeword of partial weight 2. So, the cluster size s is 4, and G contains as many clusters as there are such patterns in the Tanner graphs of $\mathcal{B}$. Note that, in order to satisfy the necessary condition on d_{min}, $\tilde{\lambda}_2$ should verify $\tilde{\lambda}_2 \leq \frac{1}{2}$.

2) Degree optimization over the Gaussian channel and choice of the permutation structure for the design rate $1/10$*:* Let us choose R to be equal to $1/10$. Also, we choose $\tilde{\lambda}_2$ to be 0.4, which is slightly less than $\frac{1}{2}$. 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 G. If the degree of clusters in G are chosen at random with $\tilde{\lambda}_2 = 0.4$, the expected values of a_i's are:

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

If one finds a structure of G with such distribution A, so that G does not contain cycles, then it fixes the permutation structure for edges connected to degree-2 variable nodes in $\mathcal{G}$. For example, let the structure of G contain the following components from Fig. 8, that we call "stars", "twigs" and "chains". Namely, we divide the Tanner graph of the base code into patterns, given in Fig.9, and associate a cluster to each of patterns. We assume that the number of clusters M is divisible by 625. The generation of $\mathcal{G}$ 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. Note that, in Table I, 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 of degree i.

Using the table, the following three points can be checked: 1) All clusters are consumed in the components, because the sum of the entries of the column corresponding to any degree i gives a_i. 2) Each entry is nonnegative. 3) The "chains" are possible to form, as the number of clusters of degree 1, used to form "chains", is even. After the degree optimization over the Gaussian channel, one obtains the following degree distribution: $\tilde{\Lambda}(x) = 0.4x + 0.264209x^2 + 0.090866x^4 + 0.236716x^8 + 0.008209x^9$.

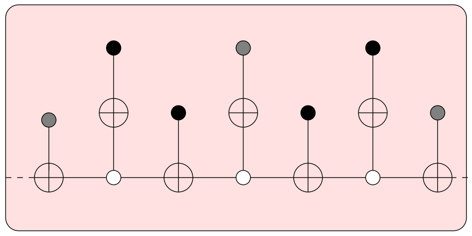

Fig. 9. Pattern in the Tanner graph of the TLDPC base code of rate $1/2$ giving rise to a cluster.

TABLE I
ARRANGING CLUSTERS TO FORM THE COMPONENTS.

	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

VI. Numerical Results and Comparison

Let us compare the error performance of designed TLDPC codes with ME-LDPC codes from [6] of rate $1/10$. Fig.10 present the word error rate (WER) of those two ensembles for lengths 6250, 18750 and 50000, over the Gaussian channel. The estimated decoding threshold of TLDPC codes is about -0.9 dB, which is around 0.4 dB away from channel capacity (-1.286 dB). ME-LDPC codes have better iterative threshold, but their error curve scales slower with the signal-to-noise ratio (SNR), and the TLDPC code starts to behave better from WER $10^{-3}, \dots, 10^{-5}$. This improvement is explained by a larger fraction λ_1 of the TLDPC code (33% against 22.5% for ME-LDPC). Note that none of two codes shows error-floors, due to their good d_{min} (*provably* linear in the multi-edge case and *conjectured* to be linear for TLDPC).

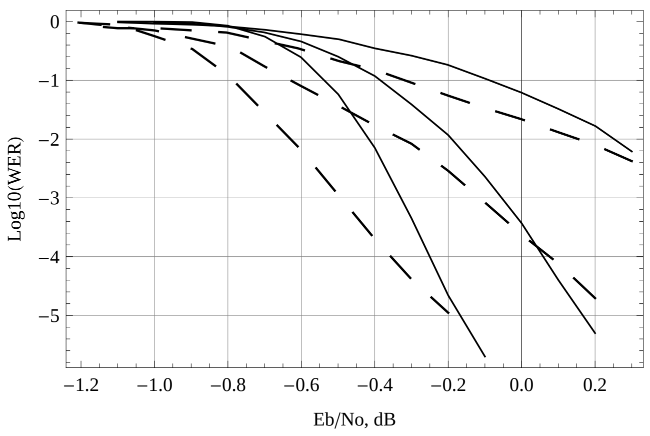

Fig. 10. Comparison of a TLDPC code of rate 1/10 (full lines) with a multi-edge LDPC code of the same rate (dashed lines) of [6]. Codelengths are 6250, 18750 and 50000.

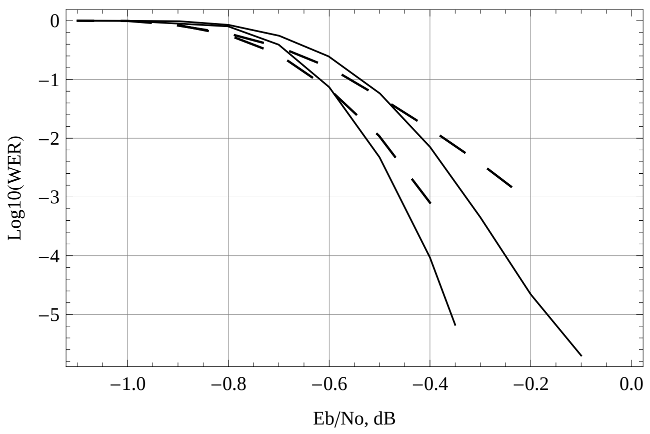

Fig. 11. Comparison of a TLDPC code of rate 1/10 (full lines) with a regular 3-layer LDGM code of rate 0.105 (dashed lines) of [9]. Codelengths are 50000 and 100000.

Let us compare the TLDPC code with a SC-LDGM code of rate 0.105, having the same variable nodes degrees as the ensemble of rate 0.1 in Table II of [9]. The comparison is presented in Fig.11. For simplicity, a constant check node degree per each class of SC-LDGM codes was taken[11], so that the obtained rates R_1, R_2 and R_3 were as close as possible to values of R_1, R_2 and R_3 in [9]. One can see that even though the SC-LDGM code has a better iterative threshold [12], its WER scales extremely slowly with SNR, which is probably due to constant d_{min}.

Finally, one can design a low-rate TLDPC ensemble of other rates than $1/10$, using the general design procedure of Section IV. For an example, Table II gives parameters of a TLDPC ensemble of rate $1/6$, compared with known codes of this rate. For the TLDPC ensemble, a base code of rate $5/8$ was taken, with $\lambda_1 = 1/4$. One can see from the table, that code ensembles with larger fractions λ_1 have better thresholds. The TLDPC ensemble has a large λ_1, so that its threshold is close to the theoretical limit (-1.703 dB), but not too large so that

[11] $d_c^1 = 20$, $d_c^2 = 5$ and $d_c^3 = 3$ in the notation of [9]

[12] Actually it is not a threshold in the usual sense since these code families have constant minimum distance.

TABLE II
COMPARISON OF CODE ENSEMBLES OF CODE RATE $1/6$.

Code ensemble of $R = 1/6$	λ_1	Threshold	d_{min}
Hybrid non-binary LDPC [12]	0	-0.41 dB	$O(\log n)$
Non-binary LDPC [13]	$2/3$	-1.073 dB	$O(\log n)$
AR3A [7]	$1/2$	-0.818 dB	$O(n^{\alpha})$, $\alpha < 1$ by [31], [8]
TLDPC with $b_{2i} = 1$	$1/4$	≈ -0.8 dB	(*)

(*)Better than $O(\log n)$, and $b_{2i+1} = 2$ (Fig.6) conjectured to be $O(n)$

d_{min} is deteriorated. Its optimized degree distribution contains $\tilde{\lambda}_2 = 0.251$ which is strictly less than $\tilde{\lambda}_2^{max} = 1/3$. Therefore, it is possible choose a permutation structure, so that G does not contain cycles.

VII. Discussion

In this paper, the following issues have been addressed:

- *A necessary condition to design sparse-graph codes with linear minimum distance in the blocklength*:
  Such a condition has been found and expressed both in terms of cycles and in terms of the average degree of the graph of codewords of partial weight 2.
- *Demonstration of the impact of degree-1 bits*:
  We have provided here an explanation of the benefits brought by degree 1 bits in the architecture by EXIT charts considerations.
- *Tradeoff between d_{min} and λ_1 and design of a new low-rate code*:
  A design procedure for low rate codes (which is well known for being a difficult issue) has been suggested. It aims at finding a tradeoff between satisfying the necessary condition on d_{min} and having a large fraction λ_1. The procedure has been applied to TLDPC codes, and a TLDPC ensemble of rate $1/10$ was designed. Its decoding performance are comparable to the one of the most efficient low-rate codes, the ME-LDPC ensemble from [6].

It is possible that the conjectured linear minimum distance property for the designed TLDPC ensemble may be proved using standard techniques based on weight distributions ([15]). The main issue to be solved in this case is to find a minimum of a multi-variable function, reflecting the structure the TLDPC construction. This would lead to a quite involved analysis, which is beyond the scope of this paper.

VIII. Acknowledgment

A part of this work was done when I.A. was with France Telecom R&D.

References

[1] I. Andriyanova, J. Tillich, and J. Carlach, "A new family of asymptotically good codes with high iterative decoding performances," in *ICC'06*, June 2006.

[2] S. Qaisar and H. Radha, "Optimal progressive error recovery for wireless sensor networks using irregular LDPC codes," in *Information Sciences and Systems, 2007. CISS '07. 41st Annual Conference on*, march 2007, pp. 232 –237.

[3] K. Li, X. Wang, G. Yue, and L. Ping, "A low-rate code-spread and chip-interleaved time-hopping uwb system," *Selected Areas in Communications, IEEE Journal on*, vol. 24, no. 4, pp. 864 – 870, april 2006.

[4] F. Tang and Z. Shi, "Iterative LDPC-Hadamard code-aided carrier frequency synchronization at extremely low SNR," in *Networks Security Wireless Communications and Trusted Computing (NSWCTC), 2010 Second International Conference on*, vol. 1, april 2010, pp. 314 –317.

[5] A. Leverrier and P. Grangier, "Unconditional security proof of long distance continuous-variable quantum key distribution," *Physical Review Letters*, vol. 102, no. 18, p. 180504, 2009.

[6] T. Richardson and R. Urbanke, "Multi-edge LDPC codes," 2005, submitted to IEEE Trans. on Inform. Theory.

[7] D. Divsalar, S. Dolinar, and C. Jones, "Low-rate LDPC codes with simple protograph structure," in *Proc. of the IEEE Int. Symp. Information Theory*, Adelaide, Australia, 2005, pp. 1622–1626.

[8] A.Otmani and J. Tillich, "On the minimum distance of generalized LDPC codes," 2008, in preparation.

[9] F. Va andzquez Arau andjo, M. Gonza andlez Lo andpez, L. Castedo, and J. Garcia-Frias, "Capacity approaching low-rate LDGM codes," *Communications, IEEE Transactions on*, vol. 59, no. 2, pp. 352 –356, february 2011.

[10] L. Ping, W. Leung, and K. Wu, "Low-rate turbo-hadamard codes," *Information Theory, IEEE Transactions on*, vol. 49, no. 12, pp. 3213 – 3224, dec. 2003.

[11] W. K. R. Leung, G. Yue, L. Ping, and X. Wang, "Concatenated zigzag-Hadamard codes," *IEEE Trans. on Information Theory*, vol. 52, no. 4, pp. 1711–1723, April 2006.

[12] L. Sassatelli and D. Declercq, "Nonbinary hybrid LDPC codes," *IEEE Trans. on Information Theory*, vol. 56, no. 10, pp. 5314 –5334, October 2010.

[13] K. Kasai, D. Declercq, C. Poulliat, and K. Sakaniwa, "Rate-compatible non-binary LDPC codes concatenated with multiplicative repetition codes," in *Proc. of the IEEE Int. Symp. Information Theory*, June 2010, pp. 844 –848.

[14] I. Andriyanova, J. Tillich, and J. Carlach, "Asymptotically good codes with high iterative decoding performances," in *Proc. of the IEEE Int. Symp. Information Theory*. IEEE, September 2005, pp. 850–854.

[15] C. Di, T.Richardson, and R. Urbanke, "Weight distributiuons of Low-Density Parity-Check codes," *IEEE Trans. on Information Theory*, vol. 52, no. 11, pp. 4839–4855, November 2006.

[16] J. Tillich and G. Zémor, "On the minimum distance of structured LDPC codes with two variable nodes of degree-2 per parity-check equation," in *Proc. of the IEEE Int. Symp. Information Theory*, Seattle, USA, 2006.

[17] J. P. Tillich, "The average weight distribution of Tanner code ensembles and a way to modify them to improve their weight distribution," in *Proc. of the IEEE Int. Symp. Information Theory*, Chicago, Illinois, 2004, p. 7.

[18] T. Richardson and R. Urbanke, "Modern coding theory," Cambridge University Press, 2008.

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

[20] N. Alon, S. Hoory, and N. Linial, "The Moore bound for irregular graphs," *Graphs Combin.*, vol. 18, pp. 53–57, 2002.

[21] M. Breiling, "A logarithmic upper bound on the minimum distance of turbo codes," *IEEE Trans. on Information Theory*, vol. 50, no. 8, pp. 1692–1710, 2004.

[22] A. Ashikhmin, G. Kramer, and S. ten Brink, "Extrinsic information transfer functions : model and erasure channel properties," *IEEE Trans. on Information Theory*, vol. 50, no. 11, pp. 2657–2673, November 2004.

[23] A. Shokrollahi, "New sequences of linear time erasure codes approaching the channel capacity," in *Proceedings of AAECC-13*, ser. Lecture Notes in Computer Science, no. 1719. Springer, 1999, pp. 65–76.

[24] J. W. Lee, "The study of turbo codes and iterative decoding," Ph.D. dissertation, Urbana, IL, 2003.

[25] J. W. Lee and R. E. Blahut, "Generalized EXIT chart and BER analysis of finite-length codes," in *Proc. IEEE Global Telecommunication Conf. (Globecom)*, San Francisco, USA, Dec. 2003, pp. 2067–2071.

[26] ——, "Convergence analysis and BER performance of finite-length turbo-codes," *IEEE Trans. Communications*, vol. 55, no. 5, pp. 1033–1043, May 2007.

[27] A. Amraoui, A. Montanari, T. Richardson, and R. Urbanke, "Finite-length scaling for iteratively decoded LDPC ensembles," *IEEE Trans. on Information Theory*, vol. 55, no. 2, pp. 497–473, Feb. 2009.

[28] I. Andriyanova, "Finite-length scaling for turbo-like ensembles on the BEC," *JSAC Special Issue on Capacity-Approaching Codes*, vol. 41, no. 6, pp. 918–927, August 2009.

[29] 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*, Nice, France, Jun. 2007, pp. 1011–1015.

[30] 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*, Toronto, Canada, Jul. 2008, pp. 1443–1447.

[31] A.Otmani, J. Tillich, and I. Andriyanova, "On the minimum distance of generalized LDPC codes," in *Proc. of the IEEE Int. Symp. Information Theory*, Nice, France, June 2007.

Appendix A
Proof of Lemma 2.1

Let n_i be the set of variable nodes of degree i, $i = 1, \ldots, s$. Thus, the length n of the constructed code and the length m of the base code are given by $n = \sum_{i=1}^{s} n_i$ and $m = \sum_{i=1}^{s} i n_i$ respectively. The minimum number k of information bits of the constructed code is then

$$k = R_b m - \sum_{i=1}^{s} (i-1) n_i = (R_b - 1)m + n, \qquad (3)$$

where $\sum_{i=1}^{s}(i-1)n_i$ is the maximum number of independent equality conditions. The rate of the constructed code is at least

$$R \overset{\text{def}}{=} \frac{k}{n} = 1 + \frac{m}{n}(R_b - 1) = 1 - (1 - R_b)\bar{\lambda}. \qquad (4)$$

This proves the lemma. □

Appendix B
Proof of Theorem 3

The area under the EXIT curve of variable nodes is ([22]):

Proposition 1: Consider BEC(p). Let λ_1 is the fraction of variable nodes of degree 1 and $\bar{\lambda}$ is the average left degree of the bipartite graph. Then the area under the EXIT chart of the variable nodes $\mathcal{A}$ is $\mathcal{A} = 1 - p\frac{\frac{1}{\bar{\lambda}} - \lambda_1}{1 - \lambda_1}$.

Proof: The area below the curve of variable nodes is

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

The area below the EXIT chart of the base code is given by a corollary of [22, Theorem 1]:

Proposition 2: Consider BEC(p). 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 curve of the base code 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 ([22]) we know that $\mathcal{A} = \frac{H(V|Y)}{(1-\lambda_i)m}$. Here V consists of a codeword of $\mathcal{B}$, 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 $H(V|Z = t) = R_b m - t$, by the assumption made on the positions of degree 1. So, $H(V|Y) = R_b m - (1-p)\lambda_1 m$, and the proposition follows immediately. □

We are ready now for the proof of Theorem 3.

Proof of Theorem 3. As long as the EXIT curve of the base code lies below the EXIT curve of variable nodes, by Propositions 2 and 1

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

Iryna Andriyanova received the M.S. Degree in electrical engineering from the National Polytechnic University of Odessa, Ukraine, in 2002, and the M.S. and Ph.D. degrees in communication systems from the National Telecommunications School of Paris (ENST), France, in 2003 and 2006 respectively. In 2007 she joined the Communications Theory Laboratory at the Federal Polytechnic School of Lausanne (EPFL), Switzerland. Since 2009 she is with the ETIS-UMR8051 group of the ENSEA/University of Cergy-Pontoise/CNRS. Iryna Andriyanova's research interests are in coding theory and wireless networks.

Jean-Pierre Tillich (M'06) was born in Mulhouse, France, in 1966. He received the Engineer degree from École des Mines de Paris, Paris, France, in 1989 and the Ph.D. degree in computer science from École Nationale Supérieure des Télécommunications (ENST), Paris, in 1994. From 1997 to 2003, he was an Assistant Professor at the University Paris XI. He is now a Researcher at the Institut de Recherche en Informatique et Automatique (INRIA), Rocquencourt, Le Chesnay, France. His research interests include classical and quantum coding theory, cryptography, and graph theory.