Noise-Aided Gradient Descent Bit-Flipping Decoders approaching Maximum Likelihood Decoding

D. Declercq, C. Winstead, B. Vasic, F. Ghaffari, P. Ivanis and E. Boutillon

***Abstract*—In the recent literature, the study of iterative LDPC decoders implemented on faulty-hardware has led to the counter-intuitive conclusion that noisy decoders could perform better than their noiseless version. This peculiar behavior has been observed in the finite codeword length regime, where the noise perturbating the decoder dynamics help to escape the attraction of fixed points such as trapping sets.**

In this paper, we will study two recently introduced LDPC decoders derived from noisy versions of the gradient descent bit-flipping decoder (GDBF). Although the GDBF is known to be a simple decoder with limited error correction capability compared to more powerful soft-decision decoders, it has been shown that the introduction of a random perturbation in the decoder could greatly improve the performance results, approaching and even surpassing belief propagation or min-sum based decoders.

For both decoders, we evaluate the probability of escaping from a Trapping set, and relate this probability to the parameters of the injected noise distribution, using a Markovian model of the decoder transitions in the state space of errors localized on isolated trapping sets.

In a second part of the paper, we present a modified scheduling of our algorithms for the binary symmetric channel, which allows to approach maximum likelihood decoding (MLD) at the cost of a very large number of iterations.

***Index Terms*—LDPC codes, noisy iterative decoding, probabilistic GDBF, noisy GDBF, MLD performance.**

I. Introduction

Recently, there has been a growing interest in studying the robustness of LDPC iterative message passing decoders, with the objective of making these algorithms tolerant to faulty gates defects. The main objective which was pursued has been to measure and predict the performance loss of iterative decoding when the algorithms are implemented with faulty hardware [1, 2, 3, 4].

Contrary to the intuitive conclusion that faulty hardware errors, modeled as transient additional noise in the algorithm, would degrade the error correction performance, it was observed that the contribution of randomness in the decoder can indeed be beneficial instead of degrading the performance. This effect has been observed mainly in the error floor region, where deterministic decoders can be stuck by the presence of trapping sets (TS), while noisy versions of the same decoders could escape from the TS attraction.

This observation has led to a new approach in LDPC decoder design. Indeed, the strength of an iterative message passing decoder has always been tackled at the local message updates, with the classical tradeoff between complexity and performance: error correction capability could be improved when more computational efforts are allocated at the variable node and check node updates. However, there is another solution to improve error correction capabililty, without changing the decoder update equations, which is by introducing noisy versions of the update equations. As a consequence, powerful iterative decoders can be implemented from very simple noisy update rules, instead of relying on heavy computations. Capitalizing on the hardware noise to espace from local attractors is tightly linked to the concept of stochastic resonnance [], and has been also observed in other contexts, i.e. for neural networks [8].

We will discuss in this work the case of the simplest and least complex iterative decoders, based on modifications of the bit-flipping (BF) algorithm [5], and study to which extend the introduction of random components in the update equations can turn them into more powerful decoders, which could compete with higher complexity decoders such as belief propagation (BP) or min-sum (MS).

Of particular interest is the gradient descent BF (GDBF) decoder [9], from which noisy versions have been derived in the recent literature [10, 11], showing strong evidence that randomness can greatly improve the performance of BF decoders. The noisy gradient descent BF (NGDBF) decoder [11], has been proposed specifically for the additive white Gaussian noise (AWGN) channel, and the probabilistic gradient descent BF decoder (PGDBF) has been proposed for the binary symmetric channel (BSC) in [10]. Through Monte Carlo simulations, it has been observed that the two noisy GDBF decoders perform much better than standard GDBF decoders, both in the waterfall and in the error floor regions, and that the extra noise can even make them compete with soft-decision message passing decoders.

In this paper, we will study in more details those two algorithms, and propose modifications on the iterative scheduling that takes advantage of the introduced randomness to greatly improve decoding performance. Since it is clear that the main interest of the extra noise is to break the attraction of fixed point of the noiseless decoders, we will focus on the simplest cases of these attractors, namely the trapping sets of the LDPC codes. We will analyse the probability of a noisy decoder to escape the attraction of a TS, when the noiseless version of the decoder is stuck. This probability of escaping a TS can serve to optimize the parameters of the random perturbation distribution. Using the outcomes of this study, we will also equip the PGDBF with a modified scheduling in order to improve the decoding performance. We show by Monte Carlo simulations that the PGDBF using the modified scheduling can surpass the most powerful noiseless decoders (BP, MS), and approach MLD performance, when a large number of decoding iterations is considered.

II. Noise-Aided Gradient Descent Bit-Flipping Decoders

A. Basic Notations for LDPC codes and BF Decoders

An LDPC code is defined by a sparse parity-check matrix H with size (M, N), where $N > M$. A codeword is a vector $x = (x_1, x_2, \dots x_N) \in \{0,1\}^N$ which satisfies $H.x^T = 0$. We denote by $y = \{y_1, y_2, \dots, y_N\}$ the output of a noisy channel, in which the bits of the transmitted codeword are perturbed by an additive memoryless noise. In this paper, we will consider the binary

D. Declercq and F. Ghaffari are with ETIS, ENSEA/University of Cergy-Pontoise/CNRS, 95014 Cergy-Pontoise, France (email: {declercq}@ensea.fr), C. Winstead is with Utah State University, Department of Electrical and Computer Engineering, UMC 4120, Logan, UT 84322, USA (email: chris.winstead@usu.edu), B. Vasic is with the Department of Electrical and Computer Engineering, University of Arizona, Tucson, USA (email: vasic@ece.arizona.edu), P. Ivanis is with School of Electrical Engineering, University of Belgrade, Serbia (email: predrag.ivanis@etf.rs), E. Boutillon is with Lab-STICC, UMR 6285, Université de Bretagne Sud, Centre de Recherche BP 92116, Lorient 56321 (email: emmanuel.boutillon@univ-ubs.fr)

This work has been funded in part by the ANR under grants ANR-15-CE25-0006-01 (NAND) and ANR-UEFISCDI joint research programme (DIAMOND), and the NSF under grants CCF-1314147 and ECCS-1500170

symmetric channel, and the additive white Gaussian noise channel. The graphical representation of an LDPC code is a bipartite graph called Tanner graph composed of 2 types of nodes, the VNs v_n, $n = 1 \dots N$ and the CNs c_m, $m = 1 \dots M$. In the Tanner graph, a VN v_n is connected to a CN c_m if $H(m,n) = 1$. Let us also denote $\mathcal{N}(v_n)$ the set of CNs connected to the VN v_n, with connection degree $d_{v_n} = |\mathcal{N}(v_n)|$, and denote $\mathcal{N}(c_m)$ the set of VNs connected to the CN c_m, with connection degree $d_{c_m} = |\mathcal{N}(c_m)|$.
The vector of variable node values at l-th iteration $\mathbf{v}^{(l)} = (v_1^{(l)}, v_2^{(l)}, \dots, v_N^{(l)})$, is referred to as a *decoder state* in l-th iteration. If $\mathbf{v}^{(l)} = \mathbf{x}$, we say that the decoder has converged to the codeword $\mathbf{x}$ in l-th iteration. A BF decoder is defined as a iterative update of the variable node values over the decoding iterations. We denote in this paper by $v_n^{(l)}$ the value of the variable node at the l-th iteration. We correspondingly denote by $c_m^{(l)}$ the binary value of the parity checks at iteration l. For each variable node v_n, we also associate an intrinsic channel value $L(y_n)^{(l)}$ which serves in the variable nodes update equations at each iteration l. The definition of $L(y_n)^{(l)}$ depends on the channel model. For the BSC channel, we have

$$L(y_n)^{(l)} = y_n \oplus v_n^{(l)} \qquad \text{[BSC]}$$

with $\oplus$ is the modulo-2 addition, while for the AWGN channel, it is defined from the log-likelihood ratio:

$$L(y_n)^{(l)} = (1 - 2\, v_n^{(l)}) \log\left(\frac{prob(y_n|v_n = 0)}{prob(y_n|v_n = 1)}\right) \qquad \text{[AWGN]}$$

The value $L(y_n)^{(l)}$ can also be used at the initialization step, in which case the first values of the VNs $v_n^{(0)}$ are simple copies of $L(y_n)^{(0)}$, with the convention that $v_n^{(0)} = 0 \ \ \forall n$.

In all variants of BF algorithms, the CN calculation can be written as

$$c_m^{(l)} = \bigoplus\nolimits_{v_n \in \mathcal{N}(c_m)} v_n^{(l)} \qquad (1)$$

In the case of GDBF algorithms, an energy function $E_{v_n}^{(l)}$ is defined for each VN based on the neighboring CN, and used to evaluate whether the value $v_n^{(l)}$ should be flipped or not. The definition of the energy function depends on the channel model.

$$E_{v_n}^{(l)} = L(y_n)^{(l)} + \sum_{c_m \in \mathcal{N}(v_n)} c_m^{(l)} \qquad \text{[BSC]} \ (2)$$

$$E_{v_n}^{(l)} = L(y_n)^{(l)} + w \sum_{c_m \in \mathcal{N}(v_n)} (1 - 2\, c_m^{(l)}) \qquad \text{[AWGN]} \ (3)$$

where w is a real valued weighting factor.
With definition (3), the real valued energy function is minimized for the bits having the lowest reliability. In [9], two modes for the bit-flipping rule at iteration l are proposed: either only the bit having lowest energy function is flipped (single flip), or a group of bits having energy function less than a predefined threshold θ are flipped (multiple flips).
For the BSC channel, the energy function is an integer and varies from 0 to $d_{v_n} + 1$. Let $b^{(l)}$ denotes the maximum energy at the l-th iteration, i.e., $b^{(l)} = \max\limits_{1 \le n \le N} (E_{v_n}^{(l)})$. The bits which have the maximum energy value are flipped. Due to the integer representation of energy function, many bits are likely to have the same maximum energy function, leading to the multiple flips mode. The fact that the number of bits to be flipped cannot be precisely controled induces a negative impact on the convergence of the GDBF algorithm for the BSC channel.

Let us further introduce the notion of flipping set $\mathcal{F}^{(l)}$, which represents the set of indices for the bits that are flipped at iteration l.

$$\mathcal{F}^{(l)} = \left\{ n \in \{1, N\}\, ;\, E_{v_n}^{(l)} = b^{(l)} \right\} \qquad \text{[BSC]} \ (4)$$

$$\mathcal{F}^{(l)} = \left\{ n \in \{1, N\}\, ;\, E_{v_n}^{(l)} \le \theta \right\} \qquad \text{[AWGN]} \ (5)$$

Using the flipping set notations, the GDBF algorithms for both the BSC and the AWGN channel can be described concisely as in algorithm 1. The algorithm is stopped when all the PCs are satisfied, or the maximum number of iterations L is reached.

Algorithm 1 Gradient Descent Bit-Flipping : **[iteration l]**

[Step 1] Compute CNs values
$c_m^{(l)}$, $\forall m = 1, \dots, M$, using (1),
[Step 2] Compute Energy functions at VNs
$E_{v_n}^{(l)}$, using (2)-(3),
[Step 3] Compute the flipping sets
$\mathcal{F}^{(l)}$, using (4)-(5),
[Step 4] Bit flipping
$\forall n \in \mathcal{F}^{(l)} \quad v_n^{(l+1)} = \overline{v_n^{(l)}}$
$\forall n \notin \mathcal{F}^{(l)} \quad v_n^{(l+1)} = v_n^{(l)}$

B. Noise Aided GDBF decoders

The principle behind the noise-aided GDBF proposed in the literature [10, 11] is to perturb the dynamics of the decoder by using randomly modified flipping sets $\widetilde{\mathcal{F}}^{(l)}$. In this paper, we name those modifications noise aided GDBF decoders (NA-GDBF).
In [10], the proposed modification is mainly analysed on the BSC channel. Based on the observation that flipping all bits that have the maximum energy could lead to a very large number of decoder failures (see section III for an example), the authors have proposed to flip only a smaller proportion of those bits, chosen at random in $\mathcal{F}^{(l)}$. The main parameter of the modification in the PGDBF is p_0, the probability that a bit flip with maximum energy is indeed realized. The randomly modified flipping set $\widetilde{\mathcal{F}}_{\text{PGDBF}}^{(l)}$ is defined as:

$$\widetilde{\mathcal{F}}_{\text{PGDBF}}^{(l)} = \left\{ n \in \mathcal{F}^{(l)}\, ;\, \epsilon_n^{(l)} < p_0 \right\} \qquad \text{[PGDBF]} \ (6)$$

where $\epsilon_n^{(l)}$ is a realization of a uniform random variable over the interval $[0, 1]$. The PGDBF modification could be applied to both the BSC and the AWGN channel using either equation (4) or (5), since the effect is simply to diminish the number of indices to be flipped.
In [11], the authors have proposed to perturb the dynamic of the GDBF using a random shift of the threshold θ by a value $\epsilon^{(l)}$ drawn at each iteration from a Gaussian distribution $\mathcal{N}(0, \sigma_p^2)$. The variance of the random shift depends on the AWGN channel variance σ: $\sigma_p = \eta\, \sigma$. As a consequence, the randomly modified flipping set $\widetilde{\mathcal{F}}_{\text{NGDBF}}^{(l)}$ becomes:

$$\widetilde{\mathcal{F}}_{\text{NGDBF}}^{(l)} \left\{ n \in [1, N]\, ;\, E_{v_n}^{(l)} \le \theta + \epsilon^{(l)} \right\} \qquad \text{[NGDBF]} \ (7)$$

Finally, the NA-GDBF are simply implemented by replacing the flipping sets $\mathcal{F}^{(l)}$ in algorithm 1 by the randomly modified flipping sets $\widetilde{\mathcal{F}}^{(l)}$.

III. Breaking Trapping Sets Attraction with NA-GDBGF

In this section, we provide specific illustrations to show that added random perturbations could help avoiding the attraction of local minima of the GDBF algorithm. The theory and detailed analysis

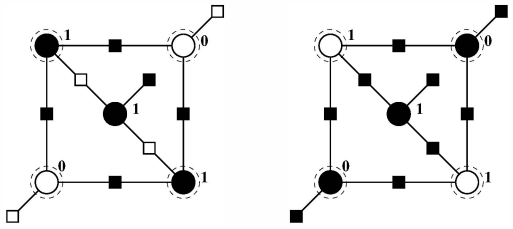
of the approach cannot be described in this paper for lack of space, and will bre reported in a future publication. Unless said otherwise, the all-zero codeword will be assumed for the analysis and the simulations.

A. Trapping sets of noiseless decoders

In deterministic BF algorithms including the GDBF, the sequence of decoder states $\{\mathbf{v}^{(l)}\}_{1\leq l\leq L}$ is completely determined by the update rule and the channel vector $\mathbf{y}$. For some channel vectors $\mathbf{y}$ the decoder does not converge to a codeword. The corresponding error pattern $\mathbf{e}$ is referred to as *uncorrectable*. In the error floor regime, the error performance is dominated by low weight uncorrectable error patterns, which are typically located in trapping sets []. In the next example, we illustrate such behavior.

In figure 1, we show a trapping set TS$(5,3)$ composed of five variable nodes, which is the smallest structure (apart from cycles) that can appear in regular LDPC codes with girth $g = 8$ and $d_v = 3$. We consider the case where only 3 errors occur on variables v_1, v_3 and v_5, while v_2 and v_4 are error free. The channel values in $\mathbf{y}$ are indicated as 0 or 1 beside the circles.

Ignoring the messages from the rest of the graph (the isolation assumption described in [6]), the GDBF decoder oscillates between the two states shown in figures 1(a) and 1(b). In this figure, white/black circles represent correct/erroneous variable nodes, while white/black squares denote satisfied/unsatisfied check nodes. At the first iteration, variable nodes v_1, v_2, v_3 and v_4 (dashed-circled) have the maximum energy $b^{(0)} = 2$, and are flipped. At the second iteration, the maximum energy is $b^{(1)} = 4$, and is associated with the same variable nodes. Thus the same set of nodes is flipped in every iteration, leading to a decoder fixed point. The above error pattern is thus uncorrectable by the GDBF algorithm.



(a) (b)

Figure 1: A trapping set (oscillating behavior) of a GDBF decoder : a) Iteration l. b) Iteration $l+1$.

It is convenient to analyse such behavior using a state space representation. Let $S = (v_1, v_2, v_3, v_4, v_5)_2$ be the state vector of the five bits composing the TS$(5,3)$. Using the example of figure 1, the GDBF decoder oscillates indefinitely between $S_{21} = (1,0,1,0,1)_2$ and $S_{26} = (0,1,0,1,1)_2$. Let us now illustrate how PGDBF and NGDBF can be analyzed on such state space representation.

B. PGDBF probability of escaping a TS

Using the example of the previous section, we can build the state space of the PGDBF decoder, with state $S_{\mathbf{e}} = S_{21}$ as root node. In principle, with a five bits state vector, the size of the state space is $2^5 = 32$ states. However, due to the particular VN and CN update equation of the PGDBF, the state space with initial state S_{21} has only 20 achievable states. Fortunately, the desired correct state S_0 is one of them.

In figure 2, we show the state space and draw the allowed transitions between the states. Each transition corresponds to a particular realization of the random noise sequence $\epsilon^{(l)} = (\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4, \epsilon_5)$. Amongst the 20 achievable states, we highlighted S_{21} which is the initial error state, S_{26} which is the oscillating state of the GDBF algorithm, S_0 which is the correct state, and S_{16} which is the state corresponding to the shortest path from S_{21} to S_0.

This shortest path is unique, and is realized using the following noise sequences $\epsilon^{(1)} = (0,1,0,1,\times)$ and $\epsilon^{(2)} = (\times,\times,\times,\times,1)$ (the $\times$ symbol means that the noise can be indifferently 0 or 1). Due to independence, the occurrence probabilities of these two configurations are $prob\{\epsilon^{(1)}\} = p_0^2(1-p_0)^2$ and $prob\{\epsilon^{(2)}\} = p_0$. The probability of correcting the 3-error pattern of figure 1 is thus $prob\{\epsilon^{(1)}\}prob\{\epsilon^{(2)}\} = p_0^3(1-p_0)^2$. Note that other sequences of flipping configurations may lead to the successful decoding in more than two iterations, but their probability of occurence will be smaller. The maximum probability of correcting this 3-error pattern is equal to 0.0346, obtained for the optimum value $p_0 = 0.6$.

The approach can be generalized to other TS and possibly averaged over all the TS present in a given LDPC codes, in order to find the best value for the parameter p_0, and predict the error correction performance of PGDBF in the error floor.

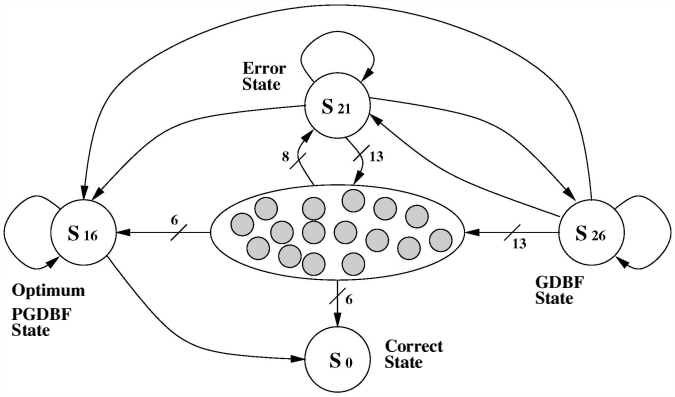


Figure 2: State space of the PGDBF decoder for the TS$(5,3)$.

C. NGDBF probability of escaping a TS

The NGDBF algorithm applies real valued perturbations directly in the energy function at each VN, hence the space of allowed decoder state transitions is much larger than for PGDBF. In particular, for the example of the TS$(5,3)$ all 32 states are achievable. Let us present how to compute the transition probabilities under the isolation assumption.

The initial errors for the AWGN channel are associated with negative channel likelihoods $L^{(0)}(y_n) < 0$. Let us assume that only the VN $(v_1, v_2, v_3, v_4, v_5)$ are initially in error, while the rest of the VNs are correct.

Using the notations of section II, and the NGDBF flipping condition of (7), the probability of flipping the bit v_n, having energy E_{v_n} is expressed as:

$$F(E_{v_n}, \theta, \sigma_p) = \frac{1}{\sqrt{2\pi}\sigma_p} \int_{-\infty}^{\theta - E_{v_n}} e^{-\frac{x^2}{2\sigma_p^2}} dx, \tag{8}$$

Since the added noise samples $\epsilon^{(l)}$ sample are independent, it is possible to compute the transition probability $\Lambda(S, S')$ between two states, S the current state before flipping and S' the state after the bit flips:

$$\Lambda(S,S') = \prod_{n\in T(S,S')} F(E_{v_n}, \theta, \sigma_p) \prod_{n\notin T(S,S')} (1 - F(E_{v_n}, \theta, \sigma_p)), \tag{9}$$

where $T(S, S')$ is the subset of indices n of $\{1, 2, 3, 4, 5\}$ where $S(n)$ and $S'(n)$ differ.

Using definition (9) for the transition probabilities in the state space, and for a given realisation of the received channel samples $\{y(n) < 0\}_{n=1\ldots5}$, it is then possible to compute the probability of correcting the TS errors from the structure of the transition matrix Λ and the Perron-Frobenius theorem. More details will be given in a future publication.

Fig. 3 shows the probability that the TS$(5, 3)$ stays in error, as a function of the iteration l, and for different values of the random perturbation variance σ_p (the value of σ_p is given as a function of a virtual signal to noise ratio SNR_p as $\sigma_p = \sqrt{10^{SNR_p/10}/2}$). This figure shows that the higher SNR$_p$, the lower the probability of error but also the slower the convergence. There is clearly a tradeoff between speed and error correction probability. In the same way as for the PGDBF case, this approach can be generalized to any TS (or a collection of them), and averaging these results over the channel noise realizations $\{L(y_n)\}$ provide optimum values for the threshold θ and the random perturbation variance σ_p.

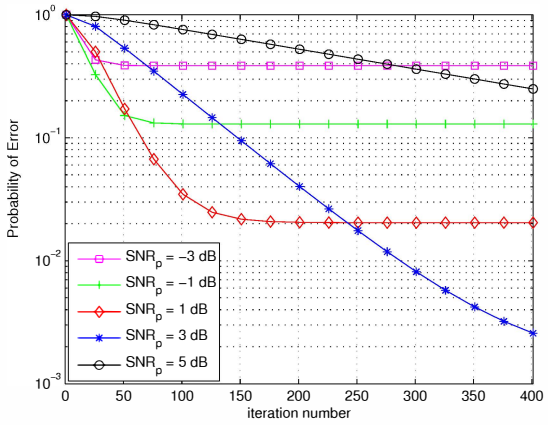


Figure 3: Probability of TS error as a function of the iteration index for several value of injected noise.

IV. PGDBF with Modified Scheduling

In this section, we further improve the performance of the PGDBF on the BSC channel by adaptively changing the scheduling in order to perturb even more the dynamics of the decoder, and avoid unwanted local attractors that are not solved by the PGDBF alone. The rationale behind the proposed modification is that some error events are still not corrected using PGDBF, i.e. there is no path in the state space that goes from the error state $S_{\mathbf{e}}$ to the correct state S_0. Our goal is to get rid of all unwanted attraction, and get performance close to maximum likelihood decoding (MLD), at the cost of a very large number of decoding iterations.

A. DDS-PGDBF with Previously Computed Threshold

The new scheduling for the PGDBF algorithm is based on using the maximum energy value computed at the *p*revious iteration $(l-1)$ to perform the bit flips of the current iteration l. We call this modified scheduling decoder-dynamic shift PGDBF (DDS-PGDBF). The new algorithm is described in Algorithm 2 for the BSC channel. The modification can be implemented using a modified definition of the flipping set, which uses the maximum value determined at the previous iteration $b^{(l-1)}$:

$$\widetilde{\mathcal{F}}^{(l)}_{\text{DDS-PGDBF}} = \left\{ n \in \{1, N\}\,;\, E^{(l)}_{v_n} \geq b^{(l-1)} \text{ and } \epsilon < p_0 \right\} \qquad (10)$$

Therefore, the computation of the maximum energy at the current iteration $b^{(l)}$ is no longer implemented in the flipping set computation, but at the end of the decoding iteration, in *S*tep 6 of the DDS-PGDBF. Note also that the condition on the energy is changed from the PGDBF as the bits considered for flipping have energy *g*reater or equal the threshold value $b^{(l-1)}$.

Algorithm 2 DDS-PGDBF : **[iteration l]**

[Step 1] Compute CNs values
$c_m^{(l)}$, $\forall m = 1, \ldots, M$, using (1),
[Step 2] Compute energy values at VNs
$E^{(l)}_{v_n}$, using (2)
[Step 3] Compute the modified flipping set
$\widetilde{\mathcal{F}}^{(l)}_{\text{DDS-PGDBF}}$ using (10),
[Step 4] Bit flipping
$\forall n \in \widetilde{\mathcal{F}}^{(l)}_{\text{DDS-PGDBF}} \quad v_n^{(l+1)} = \overline{v_n^{(l)}}$
$\forall n \notin \widetilde{\mathcal{F}}^{(l)}_{\text{DDS-PGDBF}} \quad v_n^{(l+1)} = v_n^{(l)}$
[Step 5] Update energy values at VNs
$E^{(l)}_{v_n} = E^{(l)}_{v_n} - y_n \oplus v_n^{(l)} + y_n \oplus v_n^{(l+1)}$.
[Step 6] Compute new maximum energy
$b^{(l)} = \max_{1 \leq n \leq N} (E^{(l)}_{v_n})$

The algorithm differs from PGDBF in that the flipping operation is performed *before* computing the current maximum $b^{(l)}$ and $b^{(l)}$ is found *before* considering the new CNs values. After flipping, the local energies $E^{(l)}_{v_n}$ are revised at the flipped symbol locations to account for the immediate changes due to bit flipping. The next flip threshold is then found from amongst the revised $\tilde{E}^{(l)}_v$.

This modified scheduling allows to jump in the state space of the decoder whenever it is stuck in a local minima, therefore shifting the decoder-dynamic to another trajectory. However, the jumps are not controlled in the sense that the candidate bits that form the flipping set $\widetilde{\mathcal{F}}^{(l)}_{\text{DDS-PGDBF}}$ cannot be localized. As a consequence, the probability to escape a strong local minimum is very small, and a very large number of iterations is needed to escape from them and reach MLD performance.

Note that a similar perturbation in the value of the $b^{(l)}$ for PGDBF has been proposed in [12], where the authors proposed to decrease periodically the maximum value $b^{(l)}$ to $b^{(l))-1}$ in order to perform the large jumps in the state space, and avoid the trapping set attraction. With the DDS-PGDBF, the jumps are not periodic, and follow the dynamics of the decoder.

B. Performance of DDS-PGDBF

Simulations were performed on the Tanner ($N = 155, K = 64, D_{min} = 20$) LDPC code. Fig. 4 shows the comparative frame error rate (FER) performance of DDS-PGDBF alongside the original GDBF and PGDBF algorithms, on the BSC channel with crossover probability α. For this code, the original PGDBF algorithm achieves its best performance at about 100 iterations and no longer improves with more iterations. For a maximum of 300 iterations, DDS-PGDBF shows significantly better performance. When the algorithm is allowed a very large number of iterations L, its performance improves incrementally until it approaches the MLD limit. By comparison, the classic belief propagation (BP) algorithm reaches a suboptimal performance limit after about 100 iterations. Among previously reported iterative decoders, we also plot the results of FAID decoders, which approach MLD performance by exploiting decoder diversity [7].

DDS-PGDBF is able to achieve the same result by just modifying a very simple BF decoder. It is important to note that, on average,

DDS-PGDBF terminates in a very small number of iterations due the early stopping condition, even when L is large. The average number of iterations is only 16.8 when $\alpha = 0.03$, and 4.9 when $\alpha = 0.02$, which indicates that "hard frames" that require all L iterations are very rare. Another feature of the DDS-PGDBF using a very large (or unlimited) number of iterations is that it highly likely to halt on a codeword, which is either the transmitted codeword, or an undetected error.

These performance and behaviors have been observed on other LDPC codes with various rates and lengths.

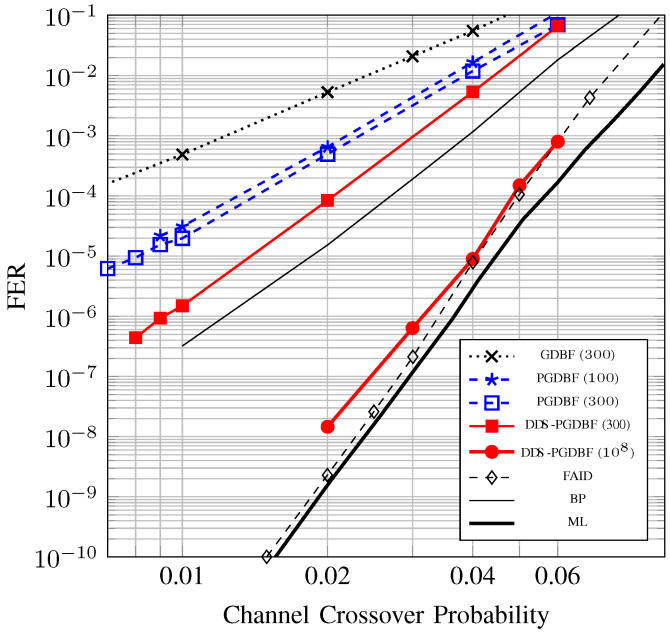


Figure 4: Frame error rate performance of different decoders on the $(N = 155,\ K = 64,\ D_{min} = 20)$ Tanner code.

V. Conclusion

In this paper, we have studied two noise-aided GDBF algorithms, and demonstrated that the random perturbations allow to escape the attraction of fixed points of the GDBF algorithms, located on trapping sets. We have also proposed a modification of the iterative scheduling that takes advantage of the introduced randomness to greatly improve decoding performance, and eventually get close to MLD performance. We believe that that this new paradigm of noise-aided hard decision LDPC decoder could become serious competitors to soft decision decoders (MS, BP), when the issue of slow convergence is solved.

References

[1] L. Varshney, "Performance of LDPC codes under faulty iterative decoding," *IEEE Transactions on Information Theory*, vol. 57, no. 7, pp. 4427–4444, July 2011.

[2] S. Yazdi, H. Cho, and L. Dolecek, "Gallager B decoder on noisy hardware," *IEEE Transactions on Communications*, vol. 61, no. 5, pp. 1660–1673, May 2013.

[3] A. Balatsoukas-Stimming and A. Burg, "Density evolution for min-sum decoding of LDPC codes under unreliable message storage," *IEEE Communications Letters*, vol. 18, no. 5, pp. 849–852, May 2014.

[4] C. Kameni Ngassa, V. Savin, D. Declercq and E. Dupraz, "Density Evolution and Functional Threshold for the Noisy Min-Sum Decoder", *in IEEE Trans. Commun.*, vol. 63, issue 5, pp. 1497–1509, May 2015.

[5] N. Miladinovic and M. Fossorier, "Improved bit-flipping decoding of low-density parity-check codes," *IEEE Transactions on Information Theory*, vol. 51, no. 4, pp. 1594–1606, 2005.

[6] S.K. Planjery, D. Declercq, L. Danjean and B. Vasic, "Finite Alphabet Iterative Decoders, Part I: Decoding Beyond Belief Propagation on the BSC", in IEEE Trans. Commun., vol. 61, issue 10, pp. 4033–4045, October 2013.

[7] D. Declercq, B. Vasic, S. Planjery and E. Li, "Finite alphabet iterative decoders approaching maximum likelihood performance on the binary symmetric channel", in the proc. of ITA workshop (invited paper), San Diego, CA, USA, February 2012.

[8] A. Karbasi, A. Salavati, A. Shokrollahi and L. Varshney, "Noise Facilitation in Associative Memories of Exponential Capacity", in Neural Computation, vol. 26, no. 11, pp. 2493–2526, November 2014.

[9] T. Wadayama, K. Nakamura, M. Yagita, Y. Funahashi, S. Usami, and I. Takumi, "Gradient descent bit flipping algorithms for decoding ldpc codes," *IEEE Transactions on Communications*, vol. 58, no. 6, pp. 1610–1614, 2010.

[10] O. A. Rasheed, P. Ivanis, and B. Vasic, "Fault-tolerant probabilistic gradient-descent bit flipping decoder," *IEEE Communications Letters*, vol. 18, no. 9, pp. 1487–1490, Sept 2014.

[11] G. Sundararajan, C. Winstead, and E. Boutillon, "Noisy gradient descent bit-flip decoding for ldpc codes," *IEEE Transactions on Communications*, vol. 62, no. 10, pp. 3385–3400, Oct 2014.

[12] B. Vasić and P. Ivaniš, "Error Errore Eicitur: A stochastic resonance paradigm for reliable storage of information on unreliable media," *IEEE Transactions on Communications* (submitted).

[13] P. Ivaniš, O. A. Rasheed, and B. Vasić, "MUDRI: A fault-tolerant decoding algorithm," *Proc. IEEE Int. Comm. Conf. (ICC 2015)*, London, UK, Jun. 8–12 2015, pp. 4291 - 4296.