Good LDPC decoders with exchanged messages of size 1, 2 and 3 bits.

Workshop on Coding for Future Optical Communications (Munich, 27 February 2018)

Emmanuel Boutillon
in collaboration with Chris Winstead, David Declercq and Franklin Cochachin

07/12/2018

Outline

- Syndrome Bit Flipping:

  BSC channel, no extrinsic, (Bit Flipping Algorithm).

- Noisy Gradient Bit Flipping Decoder:

  AWGN Channel, no extrinsic (Bit Flipping Algorithm).

- Sign-Preserving algorithm.

  AWGN Channel, message passing algorithm

Outline

- Syndrome Bit Flipping:

  BSC channel, no extrinsic, (Bit Flipping Algorithm).

- Noisy Gradient Bit Flipping Decoder:

  AWGN Channel, no extrinsic (Bit Flipping Algorithm).

- Sign-Preserving algorithm.

  AWGN Channel, message passing algorithm

Syndrome Bit Flipping

For LDPC code IEEE802.3 (size 2048, rate 0.84, $d_v = 6$, $d_c = 32$):

20 errors among 2048 bits generate in average 60 unsatisfied parities among 384.

$$\binom{2048}{20} = 10^{48} \quad \text{but} \quad \binom{384}{60} = 10^{70}!$$

Syndrome can be viewed as a compressed representation of the error pattern:

It should be possible to consider only the syndrome to suppress the error floor.

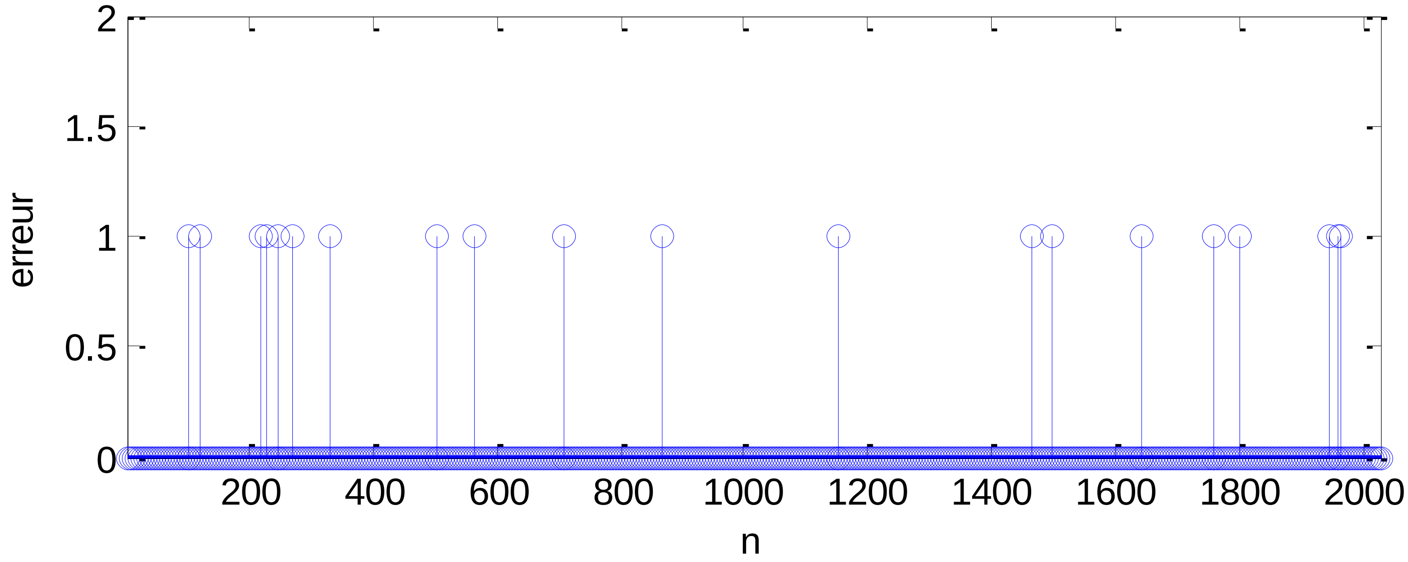
Old idea [1]: used the number of unsatisfied check connected to a variable to flip its state.

[1] Raul Benet, Adriaan De Lind Van Wijngaarden, Ralf Dohmen, Thomas Richardson, Rudiger Urbanke, "Iterative decoding of low-density parity-check (LDPC) codes", Patent EP1643653A1, 5 april 2006.

Example of SBF on a real case

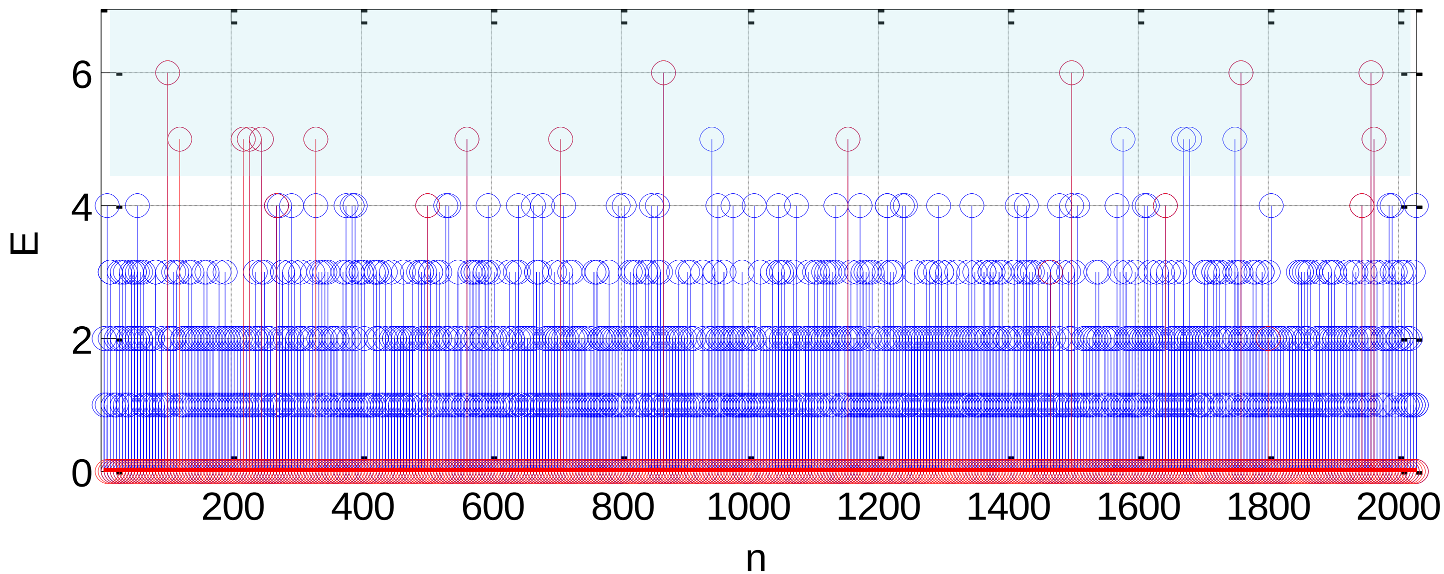
First decoder failed: the final state x doesn't respect H.x = 0.

x = c + err => S = H.x mod 2 = H(c+err) mod 2 = 0 + H.err mod 2
=> syndrom S depends only of the error vector err.



20 errors, 92 checks non fulilled (sum(S) = 92)

Value of E at the first iteration



Flip bits with E > 4 => 19 bits flipped :

14 that was in error

5 that was correct

Only 20 – 14 + 5 = 11 errors after the first iteration.

Sequence θ	=	4	4	4	3	3	
nb_err	=	20	11	6	5	3	0
nb_synd	=	92	44	22	18	14	0

Lab-STICC

cnrs

Impact of threshold sequence

Threshold sequence [4 4 4 3 3] =>

nb_err =	20	11	6	5	3	0			
nb_synd =	92	44	22	18	14	0			

Threshold sequence [4 4 4 4 4 3 3]) =>

nb_err =	20	11	6	5	5	5	3	0
nb_synd =	92	44	22	18	18	18	14	0

Threshold sequence [3 3 3 3] =>

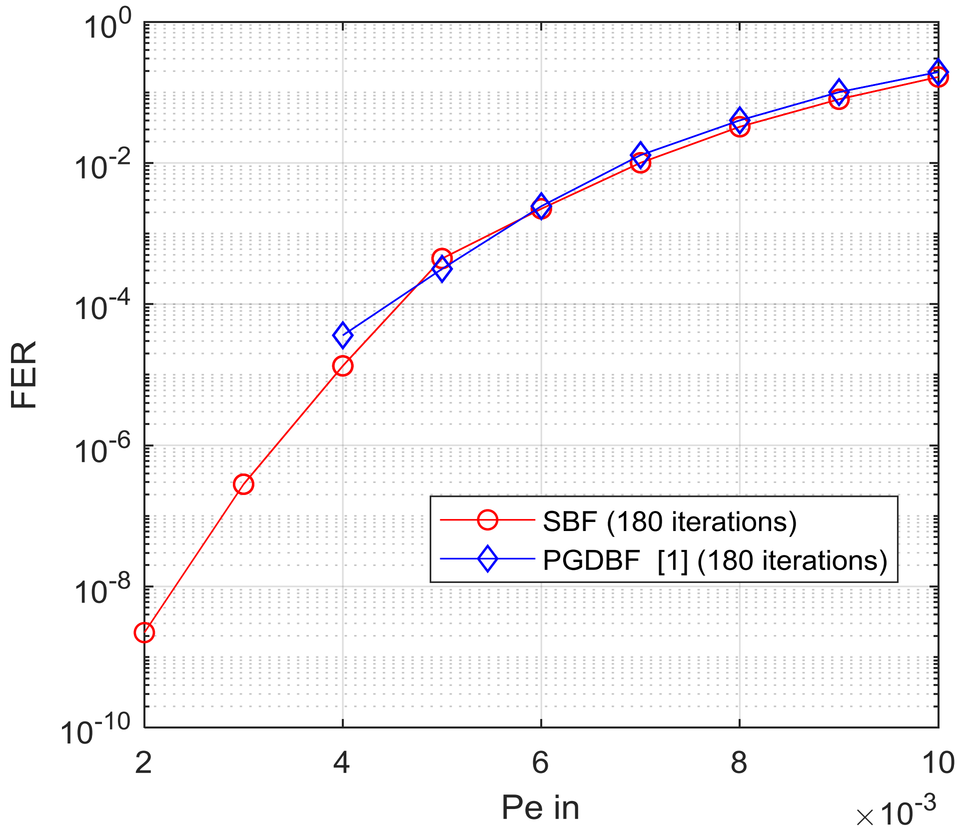
nb_err =	20	54	157	542	941
nb_synd =	92	118	172	206	202

NOTE: there is error patterns solved by [3 3 3 3] but not by [4 4 4 3 3] !

A threshold sequence is « a decoding key ».

Each key can resolve a subset of error patterns.

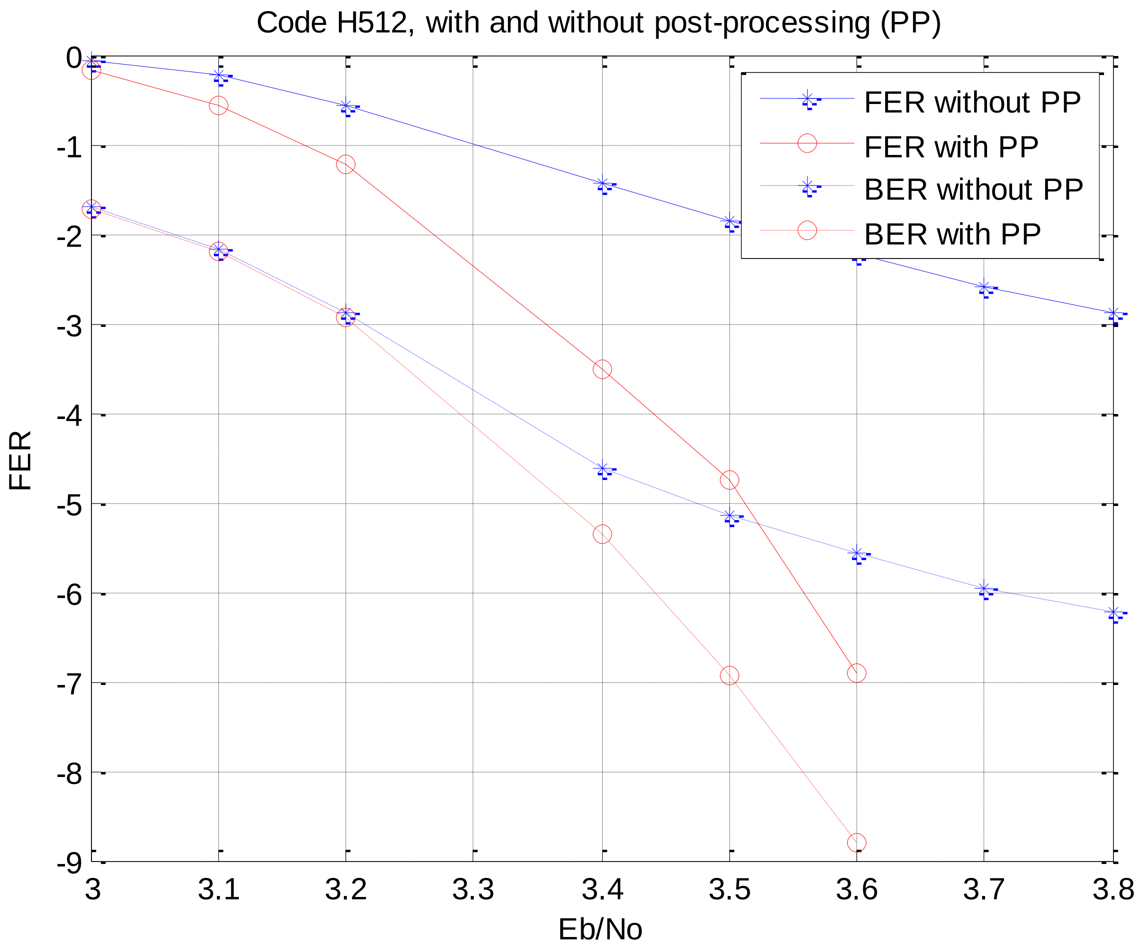
SBF alone in BSC for IEEE 802.3
$(d_v,d_c)=(6,32)$, N=2048



SBF alone can outperform PGDBF at low FER

[1] K. Le et al. "A novel high-throughput, low-complexity bit-flipping decoder for LDPC codes,"

Example of SBF Post-Processing $d_v = 5, d_c = 20, N = 10240$



Initial algorithm that converge fast toward low weigth error pattern.

Then post-processing with SBF to clear remaining error Patern.
(only 11 iterations of SBF in this example).

Outline

- Syndrome Bit Flipping:

  BSC channel, no extrinsic, (Bit Flipping Algorithm).

- Noisy Gradient Bit Flipping Decoder:

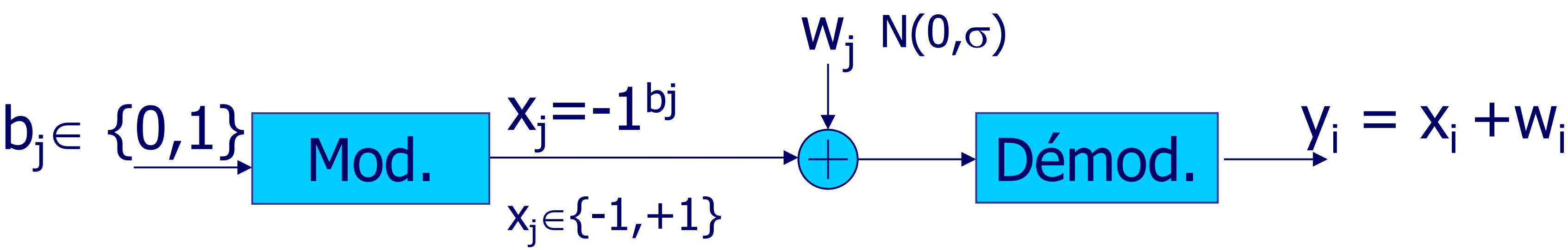
  AWGN Channel, no extrinsic (Bit Flipping Algorithm).

- Sign-Preserving algorithm.

  AWGN Channel, message passing algorithm

Optimum decoding

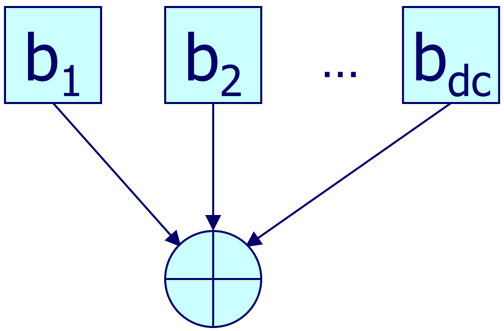
Bit of a codeword are transmitted in a noisy channel.



Decision with channel value: $x_i = \mathrm{sgn}(y_i)$

ML decoder (optimal decoder): $\hat{x} = \arg\max_{x \in C} \left\{ \sum_{i=1}^{n} x_i y_i \right\}$

Check with sign (BPSK 0=>1, 1=>-1)



$$b_1 \oplus b_2 \oplus \ldots \oplus b_{dc} = 0$$

=> Even number of bit 1

If there is no error, then, for all $j = 1 \ldots m$ $\quad S_j = \prod_{k \in N(j)} x_k = +1$

$$\hat{x} = \arg\max_{x \in C} \left\{ \sum_{i=1}^{n} x_i y_i \right\} \quad <=> \quad \hat{x} = \arg\max_{x \in C} \left\{ \sum_{i=1}^{n} x_i y_i + \sum_{j=1}^{m} S_j \right\}$$

Try to maximise the energy function $E(x_1, \ldots, x_n) = \sum_{i=1}^{n} x_i y_i + \sum_{j=1}^{m} S_j$

Gradient bit flipping algorithm

Let assume that x is not a codeword

Question: do we increase energy function flipping bit x_l?

If x_l flips to $\overline{x_l}$ then

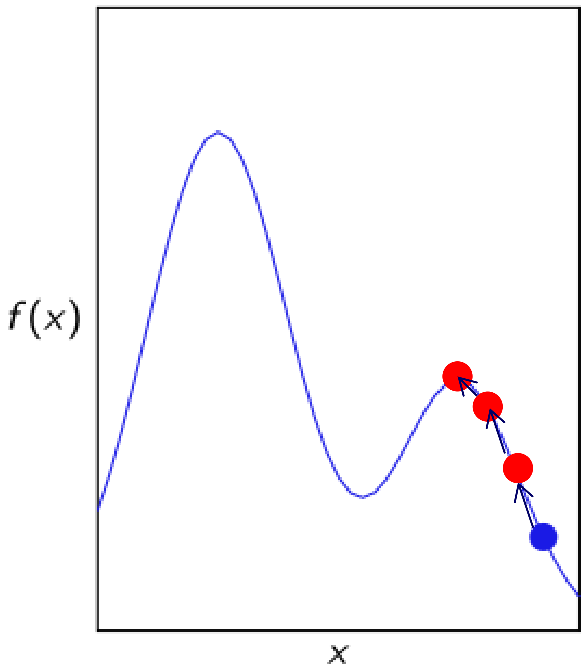
$$E_{old} \Rightarrow E_{new} = E_{old} + (\bar{x}_l - x_l)\frac{dE}{dx_l} = E_{old} - 2x_l\frac{dE}{dx_l}$$

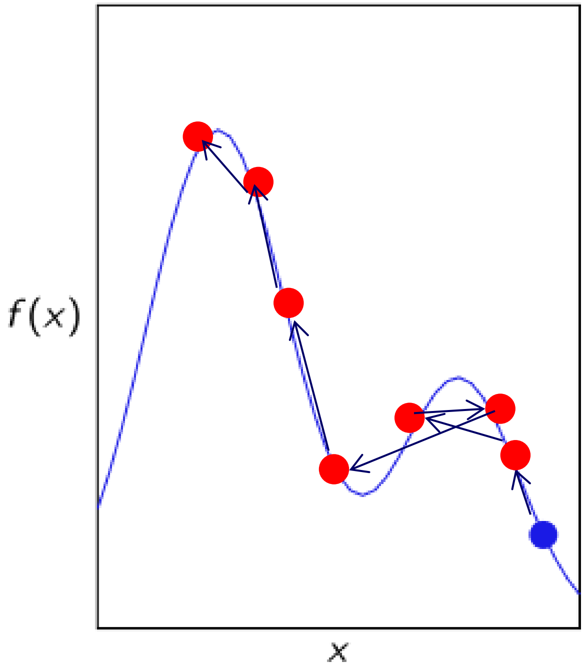
Thus, energy increases only if $E_l = x_l \frac{dE}{dx_l} < 0$.

with $E_l = x_l \frac{dE}{dx_l} = x_l y_l + \sum_{j \in N(l)} S_j$

Parallel update rule: flip the bits l verifying that $E_l < \theta$, with θ a negative threshold.

Problem with Gradient bit flipping algorithm





Problem: algorithm stucks in a local maxima.

$$E_i = x_i y_i + \sum_{j \in M(i)}^{m} S_j$$

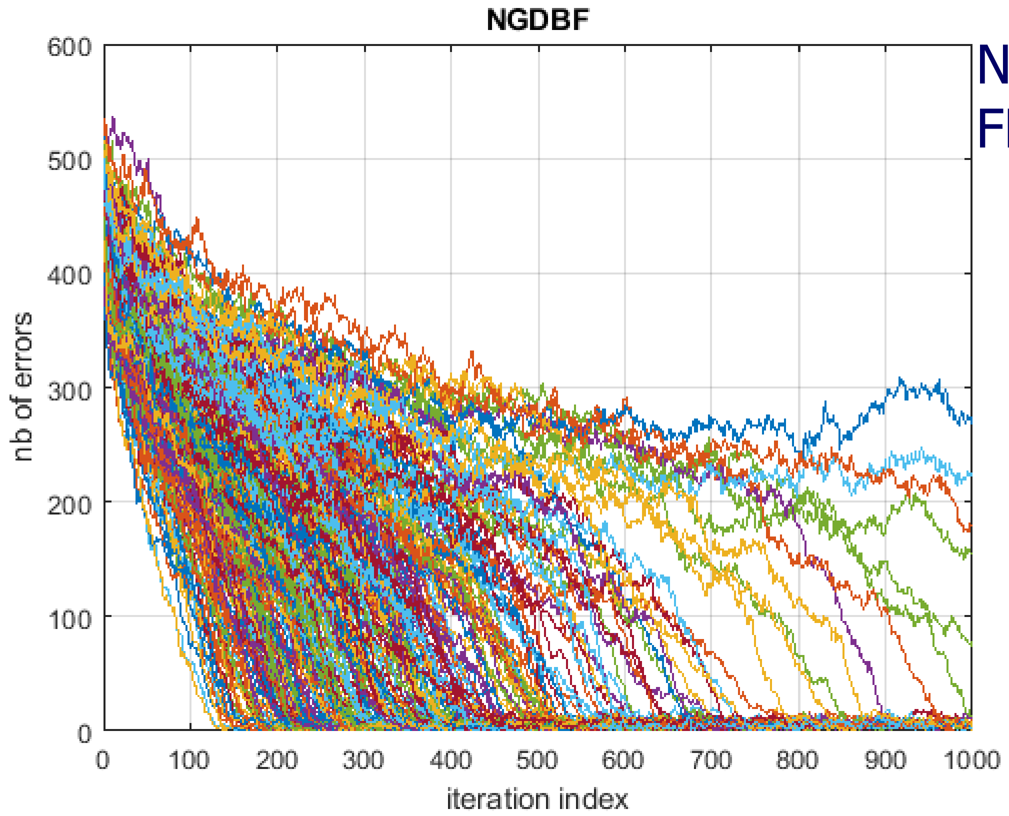
Solution to escape the local maxima: add perturbation.

$$E_i = x_i y_i + \sum_{j \in M(i)}^{m} S_j + q_i$$

AWGN

Lab-STICC

NGDBF alone



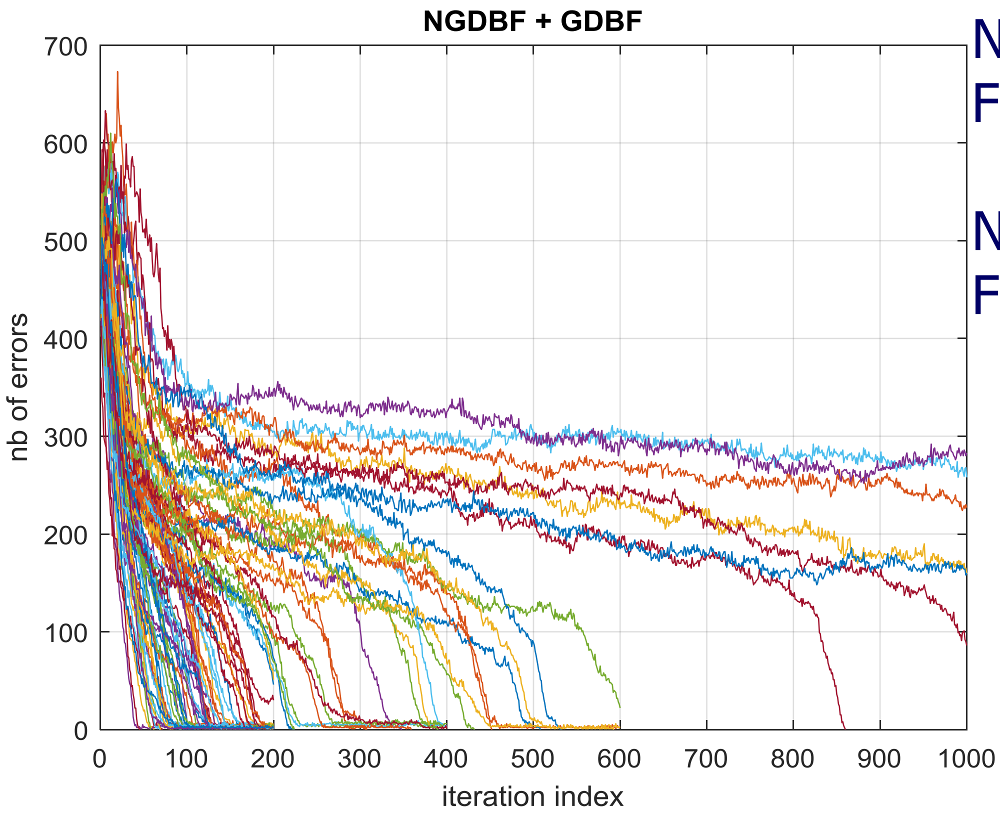
NGDBF + SBF
FER = 0.0278

(dv=5,dc = 20),
N = 10240,
Eb/N0 = 3.2 dB

Performance reference

Mixte NGDBF and GDBF

(dv=5,dc = 20),
N = 10240,
Eb/N0 = 3.2 dB



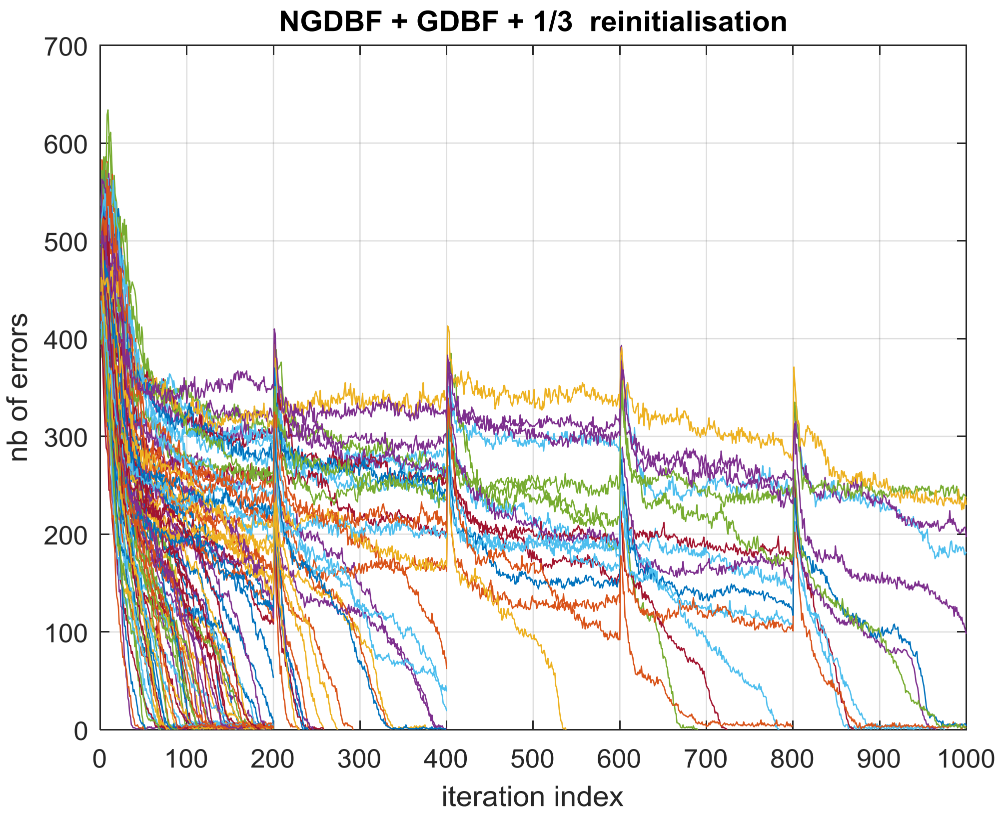
NGDBF + SBF
FER = 0.0278

NGDBF/GDBF + SBF
FER = 0.0585

Faster convergence, but FER degradation

Every 200 cycles, 1/3 of states take their initial value.

(dv=5,dc = 20),
N = 10240,
Eb/N0 = 3.2 dB



NGDBF
FER = 0.0278

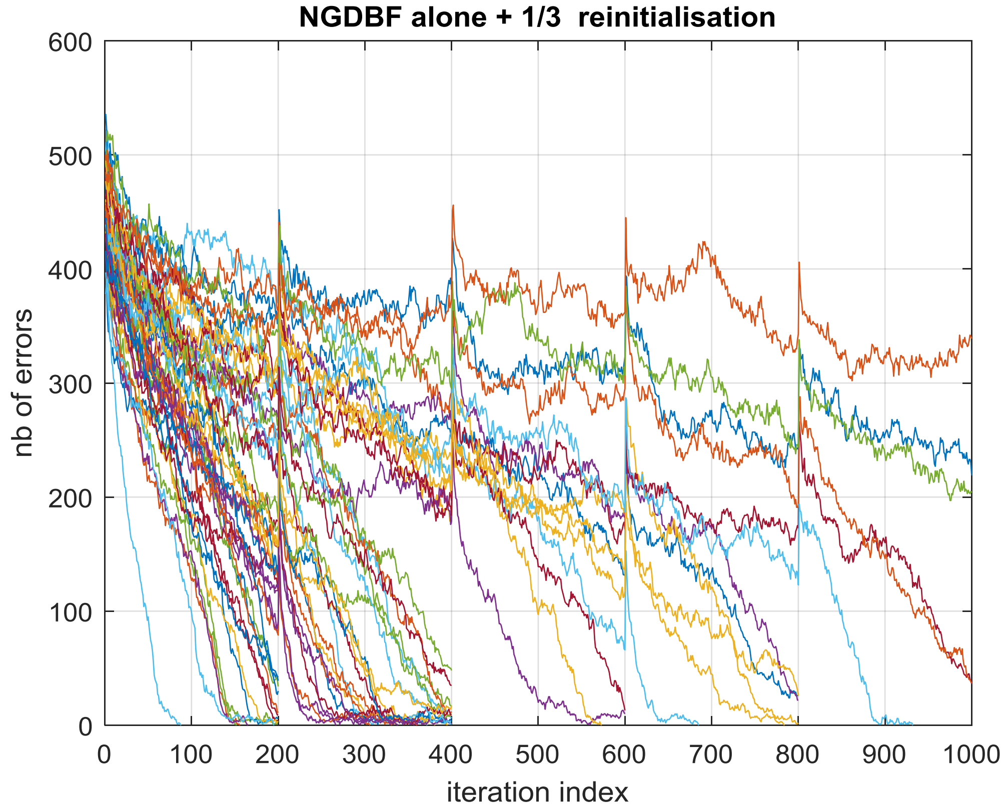
NGDBF + GDBF
FER = 0.0585

NGDBF + GDBF
+ partial restart
FER = 0.0188

Faster convergence, better performance.

Réinitialisation partielle + NGDBF simple.

(dv=5,dc = 20),
N = 10240,
Eb/N0 = 3.2 dB



NGDBF
FER = 0.0278

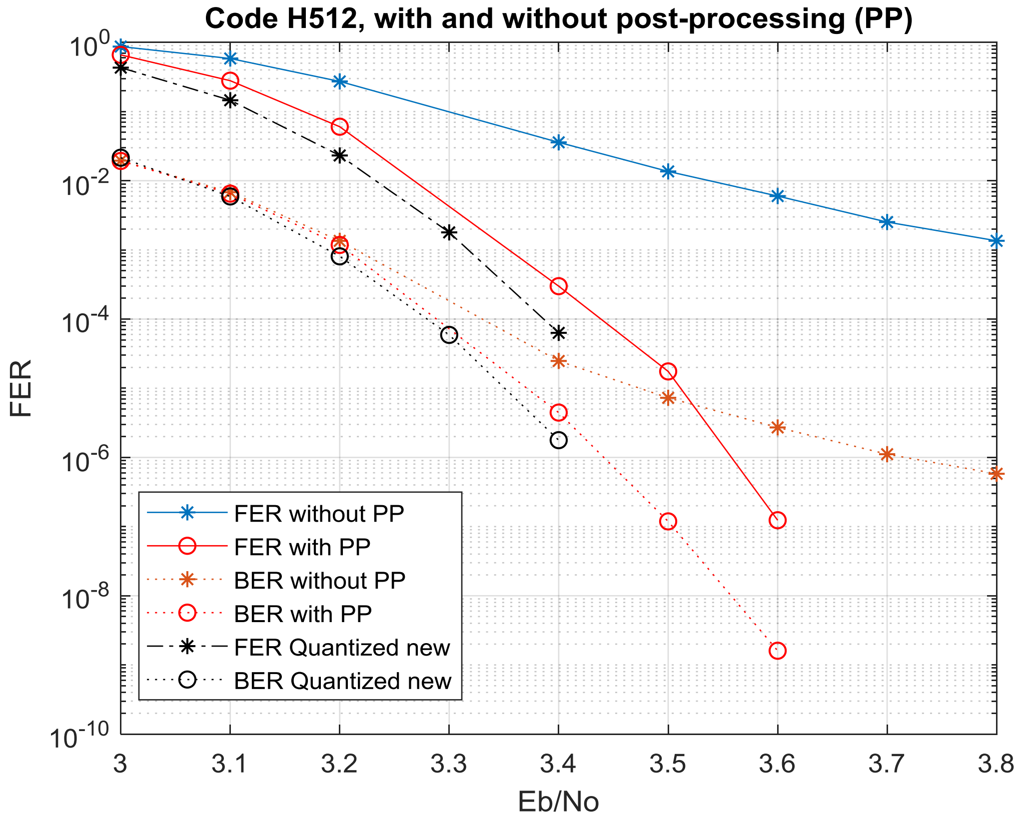
NGDBF + GDBF
FER = 0.0585

NGDBF + GDBF
+ partial restart
FER = 0.0188

NGDBF alone
+ Partial restart
FER = 0.0333

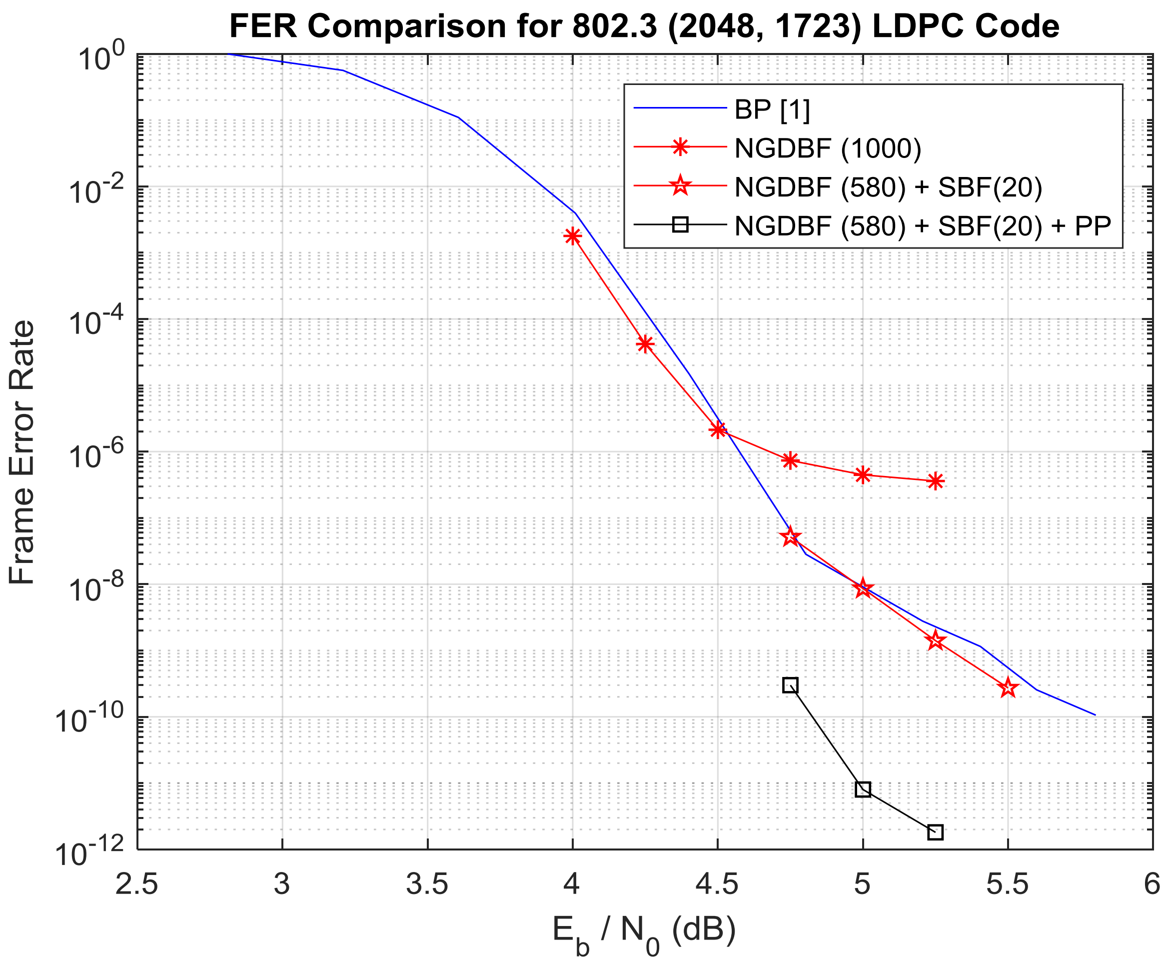
Lower performance because of lower convergence

Nouveau NGDBF
$d_v = 5, d_c = 20, N = 10240$



1000 iterations
+ SBF

NGDBF + SBF for IEEE 802.3 $(d_v,d_c)=(6,32)$, N=2048



Lab-STICC

cnrs

Outline

- Syndrome Bit Flipping:

  BSC channel, no extrinsic, (Bit Flipping Algorithm).

- Noisy Gradient Bit Flipping Decoder:

  AWGN Channel, no extrinsic (Bit Flipping Algorithm).

- Sign-Preserving algorithm.

  AWGN Channel, message passing algorithm

Outline

- Syndrome Bit Flipping:

  BSC channel, no extrinsic, (Bit Flipping Algorithm).

- Noisy Gradient Bit Flipping Decoder:

  AWGN Channel, no extrinsic (Bit Flipping Algorithm).

- Sign-Preserving algorithm.

  AWGN Channel, message passing algorithm

Oups!

Let’s us shift toward slides generated by latex…

Sign-Preserving algorithm
or "forget the zero"

Franklin Cochachin*†, Emmanuel Boutillon*, and David Declercq†

*Lab-STICC, UMR 6285, Université de Bretagne Sud, Centre de Recherche BP 92116, Lorient 56321, France
†ETIS, ENSEA/University of Cergy-Pontoise/CNRS, 95014 Cergy-Pontoise, France

NAND Project: Funded by ANR, grant number ANR-15-CE25-0006-01

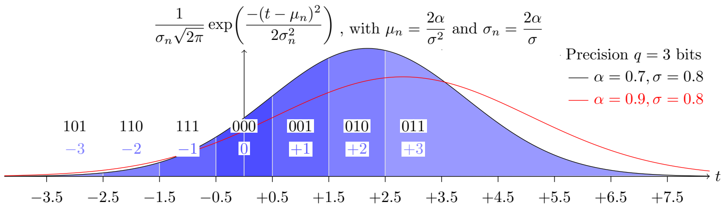
February 27, 2019

Outline

Outline

Classical OMS decoder: quantification of channel LLR

Quantified intrinsic information I_n is obtained from channel LLR y_n as $I_n = \mathcal{Q}(\alpha \times y_n)$, with $y_n \in \mathbb{R}$, α a scaling factor and $\mathcal{Q}$ a quantification rules from $\mathbb{R}$ to $\mathcal{A}_C = \{-3, -2, -1, 0, +1, +2, +3\}$ defined as $\mathcal{Q}(a) = \mathcal{S}_3(\lfloor a + 0.5 \rfloor)$, with $\mathcal{S}_3$ the saturation function toward [-3, 3].

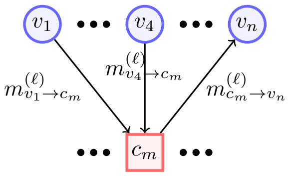


Among the 8 levels of a 3 bits representation, only 7 are used.

Low precision Offset Min-Sum Based Decoders

- **Update rule at a CNU**

$$m^{(\ell)}_{c_m \to v_n} = \left(\prod_{v \in \mathcal{V}(c_m) \backslash \{v_n\}} \mathsf{sign}\left(m^{(\ell)}_{v \to c_m}\right) \right) \times \max\left\{ \min_{v \in \mathcal{V}(c_m) \backslash \{v_n\}} \left| m^{(\ell)}_{v \to c_m} \right| - 1, 0 \right\}.$$



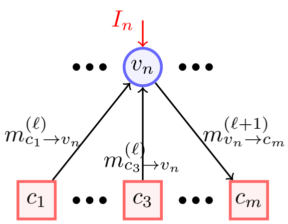
  - Note that for low precision $q = 3$ (same for $q = 4$), the offset applied in CNUs only gives us the possibility to use 5 values ($m^{(\ell)}_{c_m \to v_n} \in \{-2, -1, 0, +1, +2\}$) instead of the 7 values of $\mathcal{A}_C$.

Low precision Offset Min-Sum Based Decoders

- **Update rule at a VNU**

$$m^{(\ell+1),U}_{v_n\to c_m} = I_n + \sum_{c\in\mathcal{V}(v_n)\setminus\{c_m\}} m^{(\ell)}_{c\to v_n}.$$

exponent U indicates unsaturated message.



$$m^{(\ell+1)}_{v_n\to c_m} = \mathcal{S}_3\left(m^{(\ell+1),U}_{v_n\to c_m}\right).$$

$$m^{(\ell+1)}_{v_n\to c_m} \in \{-3,-2,-1,0,+1,+2,+3\}.$$

Low precision Offset Min-Sum Based Decoders

From the analysis of OMS-based decoders with $q = 3$

(i) the quantized LLRs $I_n \in \mathcal{A}_C = \{-3, -2, -1, 0, +1, +2, +3\}$,
(ii) v-to-c messages $m^{(\ell+1)}_{v_n \to c_m} \in \mathcal{A}_C = \{-3, -2, -1, 0, +1, +2, +3\}$, and
(iii) c-to-v messages $m^{(\ell)}_{c_m \to v_n} \in \mathcal{A}_C \backslash \{-3, +3\} = \{-2, -1, 0, +1, +2\}$.

The OMS-based decoders are suboptimal

It can be clearly noted that all combinations that can be obtained from $q = 3$ bits is not used.

How to use the 8 quantization levels for precision $q = 3$

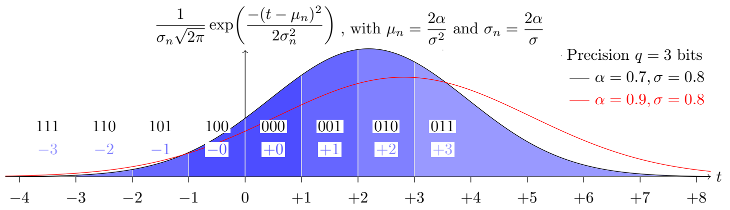
We define a new decoder called Sign-Preserving decoder where:

- The quantization of LLRs has to be changed.
- CNU has to be changed.
- VNU has to be changed.

Outline

Quantization used for Sign-Preserving Min-Sum Decoders

- In order to use the 8 levels of $q = 3$ bits, we define the message alphabet as $\mathcal{A}_S = \{-3, -2, -1, -0, +0, +1, +2, +3\}$. Using the sign-and-magnitude representation we have $100_2 = -0$, $000_2 = +0$, etc.
- A new quantification process $\mathcal{Q}^*$ is used: $I_n = \mathcal{Q}^*(\alpha \times y_n) \in \mathcal{A}_S$ where $\mathcal{Q}^*(a) = (\text{sign}(a), \mathcal{S}_3(\lceil \alpha \times |a| \rceil - 1))$

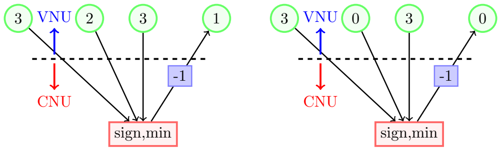


The *channel gain factor* α represents a degree of freedom in the decoder definition.

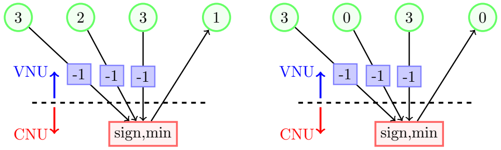
CNU of Sign-Preserving Min-Sum Decoders

The update rule at a CNU can be written in two equivalent ways.

Classical method: $\left|m^{(\ell)}_{c_m \to v_n}\right| = \max\left\{ \min_{v \in \mathcal{V}(c_m)\setminus\{v_n\}} \left|m^{(\ell)}_{v \to c_m}\right| - 1, 0 \right\}.$



Equivalent method: $\left|m^{(\ell)}_{c_m \to v_n}\right| = \min_{v \in \mathcal{V}(c_m)\setminus\{v_n\}} \left(\max\left(\left|m^{(\ell)}_{v \to c_m}\right| - 1, 0 \right) \right).$



Therefore, we can move the offset from CNs to VNs.

VNU of Sign-Preserving Min-Sum Decoders

Moving the offset from CNs to VNs, the update rule at a VNU of OMS-based decoders can be rewritten as

$$m^{(\ell+1),U}_{v_n \to c_m} = I_n + \sum_{c \in \mathcal{V}(v_n) \backslash \{c_m\}} m^{(\ell)}_{c \to v_n}.$$

The offset is subtracted before saturation to allow -3 and 3 values in the variable to check message.

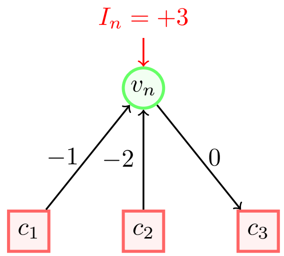
$$m^{(\ell+1)}_{v_n \to c_m} = \mathsf{sign}\left(m^{(\ell+1),U}_{v_n \to c_m}\right) \times \mathcal{S}_3\left(\max\left(\left|m^{(\ell+1),U}_{v_n \to c_m}\right| - 1, 0\right)\right)$$

Problem of the update rule at VNUs

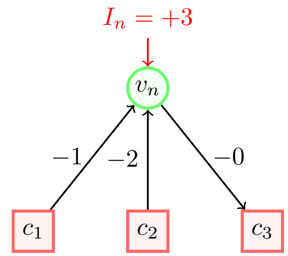
The v-to-c message $m^{(\ell+1)}_{v_n \to c_m}$ can be zero, zero does not have any information about the bit value, this means that the VNU can erase the bit value.

VNU of Sign-Preserving Min-Sum Decoders

Classical VNU:



Modified VNU:



Idea to avoid the zero value.

- **Modified VNU:** we propose to use half the sum of the signs of the c-to-v messages and the sign of the quantized LLR
  Example: $a = (+\mathbf{3}-\mathbf{1}-\mathbf{2}) + (+\mathbf{1}-\mathbf{1}-\mathbf{1})/2 = -\mathbf{0.5}$, hence $m^{(\ell),U}_{v_n \to c_3} = (\mathrm{sign}(a), \lfloor |a| \rfloor) = -\mathbf{0}$ and $m^{(\ell)}_{v_n \to c_3} = -\mathbf{0}$.

We define the Sign-Preserving Min-Sum decoders as decoders that always preserve the sign of the messages [1].

Update rules for SP-MS Decoders

Let us denote by $\mu_{v_n \to c_m}^{(\ell)}$ the *sign-preserving factor* of the message $m_{v_n \to c_m}^{(\ell+1)}$, defined as

$$\mu_{v_n \to c_m}^{(\ell)} = \xi \times \mathsf{sign}(I_n) + \sum_{c \in \mathcal{V}(v_n) \setminus \{c_m\}} \mathsf{sign}\left(m_{c \to v_n}^{(\ell)}\right),$$

where $\xi = \begin{cases} 0, & d_v = 2, \\ 1, & d_v \in \{3, 5, 7, ...\}, \\ 2, & d_v \in \{4, 6, 8, ...\}. \end{cases}$

By construction $\mu_{v_n \to c_m}^{(\ell)}$ take its value between $\{-1, 1\}$ for $d_v = 2$, $\{-d_v, -d_v + 2, \ldots, -1, 1, \ldots, d_v\}$ for d_v odd and $\{-d_v - 1, -d_v + 1, \ldots, -1, 1, \ldots, d_v + 1\}$ for d_v even.

Update rules for SP-MS Decoders

- Let us redefined $m_{v_n \to c_m}^{(\ell+1),U}$ as

$$m_{v_n \to c_m}^{(\ell+1),U} = \frac{\mu_{v_n \to c_m}^{(\ell)}}{2} + I_n + \sum_{c \in \mathcal{V}(v_n) \setminus \{c_m\}} m_{c \to v_n}^{(\ell)}$$

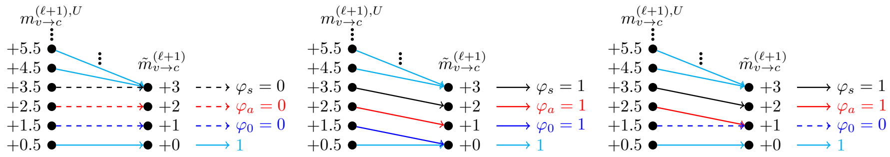
- $m_{v_n \to c_m}^{(\ell+1),U}$ takes its value in the set $\mathcal{A}_U = \{\ldots, -1.5, -0.5, +0.5, +1.5, \ldots\}$.
- and we obtain a message $m_{v_n \to c_m}^{(\ell+1)}$ that has always a defined sign as

$$m_{v_n \to c_m}^{(\ell+1)} = \left(\mathsf{sign}\left(m_{v_n \to c_m}^{(\ell+1),U}\right), \mathcal{S}_3\left(\mathsf{max}\left(\left\lfloor\left|m_{v_n \to c_m}^{(\ell+1),U}\right|\right\rfloor - 1, 0\right)\right)\right).$$

Outline

Optimization of Sign-Preserving Min-Sum Decoders

Thanks to density evolution, we can assess the optimal saturation rules at variable node level. For example, for $q_{ch} = 3$ (channel quantization), $q = 3$ (message quantization), $(d_v, d_c) = (4, 8)$ we obtained:



(a) SP-MS. (b) SP-OMS. (c) Optimized SP-MS.

MS Decoder	OMS decoders	Optimized SP-MS	DE gain	SNR gain
$\delta_{db} = 2.736$	$\delta_{db} = 2.322$	$\delta_{db} = 1.982$	**0.3399**	**0.32**

SNR gains in the waterfall correspond to what was predicted with DE.

SP-MS decoders for (5,20)-regular LDPC codes

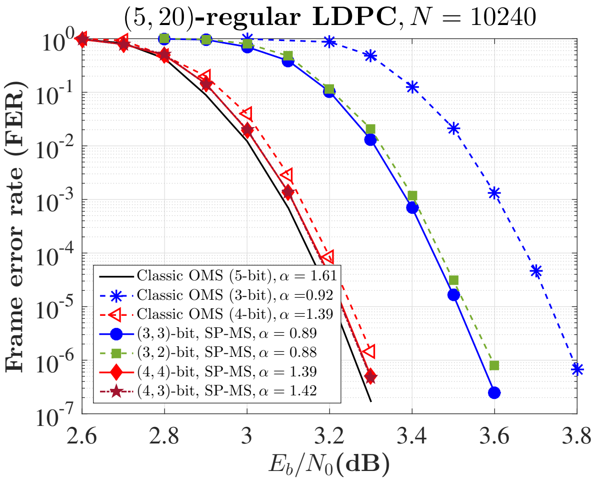


Figure: FER performance of SP-MS decoders for precision $q \in \{2, 3, 4\}$.

SP-MS decoders for (6,32)-regular LDPC codes

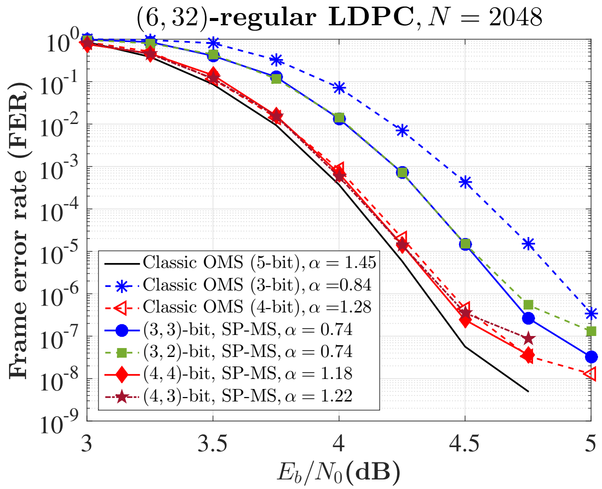


Figure: FER performance of SP-MS decoders for precision $q \in \{2, 3, 4\}$.

SP-MS decoders for (4,64)-regular LDPC codes

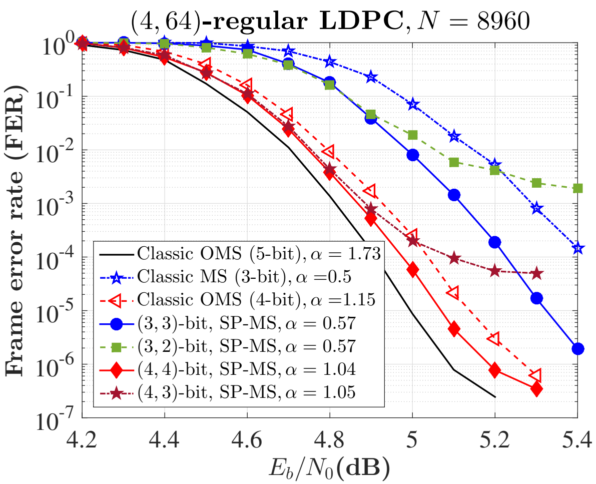


Figure: FER performance of SP-MS decoders for precision $q \in \{2, 3, 4\}$.

SP-MS decoders for (3,18)-regular LDPC codes

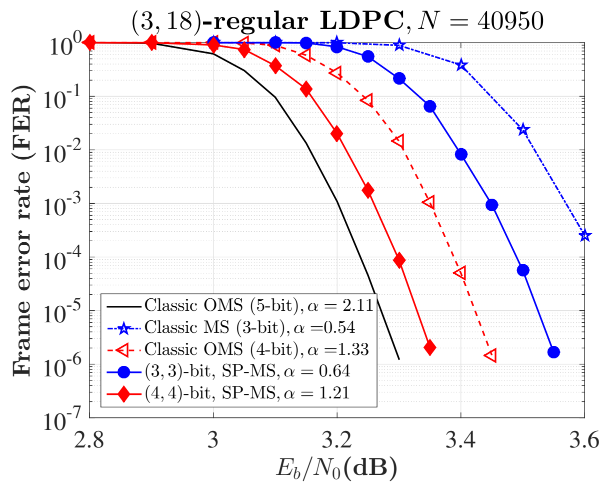


Figure: FER performance of SP-MS decoders for precision $q \in \{3, 4\}$.

Qualitative result obtained by SP-MS algorithm

For $(q_{ch}, q) = (4, 4)$, SP-MS gives small gain compared to MS or OMS

For $(q_{ch}, q) = (4, 3)$, SP-MS is almost equivalent to SP-MS with $(q_{ch}, q) = (4, 4)$!

For $(q_{ch}, q) = (3, 3)$, SP-MS gives 0.2 up to 0.4 dB of gain compared to MS or OMS.

For $(q_{ch}, q) = (3, 2)$, SP-MS is almost equivalent to SP-MS with $(q_{ch}, q) = (3, 3)$!.

Outline

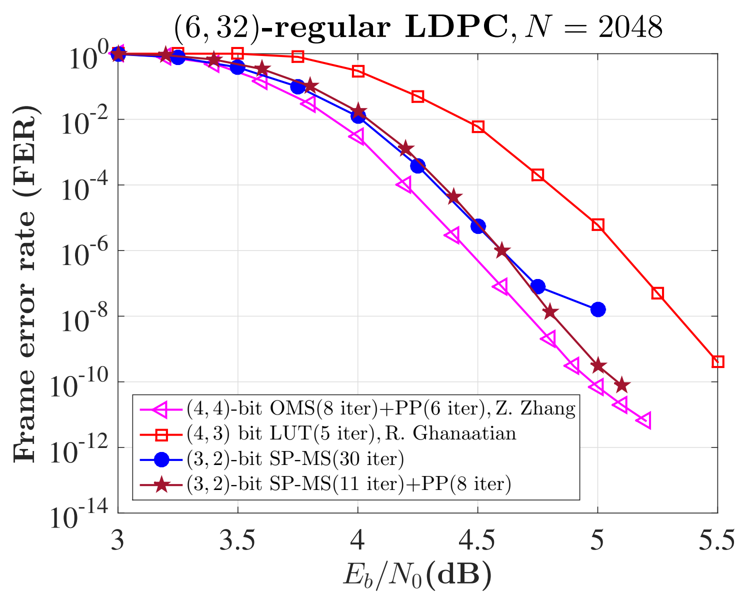
Synthesis results for regular LDPC Codes

IEEE 802.3 LDPC code: Rate 0.8143 $(2048, 1723)$ with regular $(6, 32)$ degree distribution, quantification (3,3).

	This work	Ghanaatian,2018,[2]	Zhang,2010,[3]
Technology	28nm FDSOI	28nm FDSOI	65nm CMOS
Decoder	SP-MS	finite-alphabet	OMS
Quantization	3 bits	3 bits	4 bits
Iterations	9+6	5	8+6
Architecture	full-parallel	unrolled full-parallel	partial-parallel
E_b/N_0 @ FER= 10^{-10}	5.03 dB	5.51 dB	4.98 dB
Frequency	500 MHz†	862 MHz	700 $\xrightarrow{28nm}$ 1000 MHz
Core area	2.56 mm^{2†}	16.2 mm^2	5.05 $\xrightarrow{28nm}$ 1.77 mm^2
Throughput	68.3 Gbit/s†	588 Gbit/s	13.3 $\xrightarrow{28nm}$ 19 Gbit/s
Hardware efficiency	26.7 Gbit/s/mm^{2†}	36.3 Gbit/s/mm^2	2.63 $\xrightarrow{28nm}$ 10.7 Gbit/s/mm^2
Throughput (4.5 dB)	256 Gbit/s/mm^{2†}	588 Gbit/s/mm^2	33.3 $\xrightarrow{28nm}$ 48.2 Gbit/s/mm^2
Hardware efficiency (4.5 dB)	100 Gbit/s/mm^{2†}	36.3 Gbit/s/mm^2	6.59 $\xrightarrow{28nm}$ 26.91 Gbit/s/mm^2

† Preliminary results.

Emulation on FPGA for the IEEE 802.3 ETHERNET code



Outline

Conclusions

We believe that NGDBF and SP-MS can reach 200 Gbit/s/mm2 on 28 nm technology with very good FER.

Could be very interesting to assess the algorithm performance for a convolutional LDPC code used in optical fiber.

Tera bit/s decoding throughput with ultra low FER should be feasible in an ASIC with low energy per bit.

Thank you for listening!

Questions?

References I

[1] F. Cochachin, E. Boutillon, and D. Declercq, "Optimization of sign-preserving noise-aided min-sum decoders with density evolution", in *2018 IEEE 10th International Symposium on Turbo Codes Iterative Information Processing (ISTC)*, Dec. 2018, pp. 1–5. DOI: 10.1109/ISTC.2018.8625309.

[2] R. Ghanaatian, A. Balatsoukas-Stimming, T. C. Müller, M. Meidlinger, G. Matz, A. Teman, and A. Burg, "A 588-gb/s ldpc decoder based on finite-alphabet message passing", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 2, pp. 329–340, Feb 2018, ISSN: 1063-8210. DOI: 10.1109/TVLSI.2017.2766925.

[3] Z. Zhang, V. Anantharam, M. J. Wainwright, and B. Nikolic, "An efficient 10gbase-t ethernet ldpc decoder design with low error floors", *IEEE Journal of Solid-State Circuits*, vol. 45, no. 4, pp. 843–855, April 2010, ISSN: 0018-9200. DOI: 10.1109/JSSC.2010.2042255.