Bubble check: simplified check node architecture for non-binary LDPC

E. Boutillon, L. Conde-Canencia

The Extended Min Sum (EMS) algorithm has recently been proposed for non-binary LDPC decoding. In this letter, we present a simplified version of the EMS check node update, named bubble check. This novel technique can reduce the number of comparison operations by a factor of 3, resulting in a lower hardware complexity, without introducing any significant perfomance degradation.

Introduction: Non-binary Low-Density Parity-Check (NB-LDPC) codes are constructed as a set of parity equations over a Galois Field GF(q). They are known to be an efficient alternative to binary LDPC for the transmission of short frames. However, high decoding complexity of NB-LDPC codes (especially the check node processors) remains a bottleneck in their hardware realization. In this letter we propose a low complexity algorithm named Bubble Check to perform efficiently the check node processing.

Check node processing in NB-LDPC decoders: Using the forward-backward algorithm, a check node of degree d_c can be decomposed in $3(d_c\text{-}2)$ Elementary Check Nodes (ECN) where an ECN has two input messages **U** and **V** and one output message **E**. Each message **U** contains the probability distribution of the associated variable u, i.e. the probability density function **U** defined by the q probabilities $\{P(u=\alpha)\}_{\alpha\in \mathrm{GF}(q)}$. In order to verify the elementary check equation, u, v and e must verify the parity equation: $u+v+e=0$, where "+" stands for the addition in GF(q). Assuming that **U** and **V** are known, the extrinsic information **E** can be computed for every value γ of GF(q) as [1]:

$$P(e=\gamma)=\sum_{(\alpha,\beta)\in GF(q)^2/\alpha+\beta=\gamma} P(e=\alpha)\times P(v=\beta) \tag{1}$$

The direct computation of (1) requires q^2 multiplications and additions. Performing the computation in the frequency domain reduces the computational complexity to $q.\log(q)$ multiplications and additions [2]. The complexity of (1) can be further reduced by *a*) only considering the n_m highest values of vectors **U**, **V** and **E**; *b*) performing the operation using the Log-Likelihood Ratio (LLR) $\mathrm{LLR}_U(\alpha)=\mathrm{Log}(\mathrm{P}(u=\alpha)/\mathrm{P}(u=\alpha_0))$ where α_0 is the GF(q) value with the highest probability and *c*) approximating the addition in the probability domain by the maximum in the LLR domain. These are the simplifications considered in the EMS algorithm [3]. Equation (1) then becomes:

$$\mathrm{LLR}_E(\gamma)=\underset{\alpha\in\mathbf{U}^{gf}(n_m),\beta\in\mathbf{V}^{gf}(n_m)/\alpha+\beta=\gamma}{MAX} \mathrm{LLR}_U(\alpha)+\mathrm{LLR}_V(\beta) \tag{2}$$

where $\mathbf{U}^{gf}(n_m)$ (respectively $\mathbf{V}^{gf}(n_m)$) is the set of the n_m GF(q) values corresponding to the highest LLR values of **U** (respectively **V**). Since the LLR are all negative, it is more convenient to consider $\mathrm{LLR_m}$ = -LLR to deals with positive values. Note that using $\mathrm{LLR_m}$ instead of LLR replaces the MAX operator in equation (2) by the MIN operator. In the rest of the paper, $\mathbf{U}(i)$ will denote the i^{th} smallest $\mathrm{LLR_m}$ value and $\mathbf{U}^{gf}(i)$ the associated GF(q) symbol (*i.e.* $\mathbf{U}(i)$ = $\log(\mathrm{P}(u=\mathbf{U}^{gf}(0)=\alpha_0)-\log(\mathrm{P}(u=\mathbf{U}^{gf}(i)))$. Thus, the values $\mathbf{U}(i)_{i=1..nm}$ verify $\mathbf{U}(1)=0\leq \mathbf{U}(2)\leq\ldots\leq \mathbf{U}(n_m)$. The same convention holds also for **V** and **E**.

From **U** and **V**, it is possible to generate the $n_m\times n_m$ matrix $\mathbf{T_\Sigma}$ defined as $\mathbf{T}_\Sigma(i,j) = \mathbf{U}(i)+\mathbf{V}(j)$, for $(i,j)\in[1, n_m]^2$. As **U** and **V** are sorted in increasing order, $\mathbf{T_\Sigma}$ satisfies:

Theorem 1: $\forall (i,j) \in [1, n_m]^2$, $\forall (i',j') \in [1, n_m]^2$, $i \leq i'$ and $j \leq j'$ => $\mathbf{T_\Sigma}(i,j) \leq \mathbf{T_\Sigma}(i',j')$

In other words, the lines and the columns of $\mathbf{T_\Sigma}$ are still sorted in increasing order. Using this property, the EMS tries to extract the n_m lowest values of $\mathbf{T}_\Sigma$ corresponding to n_m distinct GF(q) symbols as described by the following.

The EMS algorithm

1. Initialization: $\mathbf{E} = \varnothing$ and initialize the set of n_m competing candidates, or competing bubbles (CB), with the first column $\mathbf{T}_\Sigma(i,1)$ of $\mathbf{T}_\Sigma$.
2. FOR k=1 to n_{op} LOOP
    a. Extract the lowest value of CB $\mathbf{T}_\Sigma(i,j)$ and compute the associated GF(q) value $\gamma(i,j)= \mathbf{U}^{gf}(i)+\mathbf{V}^{gf}(j)$.
    b. Replace the extracted value in CB by $\mathbf{T}_\Sigma(i,j+1)$.
    c. IF $\gamma(i,j)\notin \mathbf{E}^{gf}$, then add the new candidate to the set **E**, EXIT loop if card($\mathbf{E}$)=n_m.

The value n_{op} is the number of allowed trials to find n_m distinct GF(q) values in **E**, thus $n_{op} \geq n_m$. If n_{op} is large enough, then equation (2) will be exactly computed. In order to extract the lowest value of CB in one clock cycle, [3] proposes to perform a parallel insertion of the new incoming value in CB. This operation requires n_m comparisons in parallel. Since n_{op} cycles are needed, the global complexity of the EMS algorithm is then $n_{op}\times n_m$ comparisons.

The bubble check principle: As $\mathbf{T}_\Sigma(1,1)$ is the minimum value of $\mathbf{T}_\Sigma$, $\mathbf{E}(1) = \mathbf{T}_\Sigma(1,1)$ can be extracted directly. Similarly, the second lowest value of $\mathbf{T}_\Sigma$ is either $\mathbf{T}_\Sigma(2,1)$ or $\mathbf{T}_\Sigma(1,2)$, i.e. there are only two possible candidates for $\mathbf{E}(2)$. The third element $\mathbf{E}(3)$ can be either between $\mathbf{T}_\Sigma(3,1)$ and $\mathbf{T}_\Sigma(1,2)$ or between $\mathbf{T}_\Sigma(1,3)$ and $\mathbf{T}_\Sigma(2,1)$ according to the selected second smallest value i.e. $\mathbf{T}_\Sigma(2,1)$ or $\mathbf{T}_\Sigma(1,2)$ respectively. A more general question is: what is the maximum number n_b of competing bubbles that needs to be checked in order to extract the n^{th} lowest value? This problem can be answered by reversing the question: assuming n_b bubbles, what is the maximum number of lowest values that can be extracted from $\mathbf{T_\Sigma}$? When the n_b bubbles form a diagonal $\{\mathbf{T_\Sigma}(k, n_b-k+1)\}_{k=1..nb}$ in $\mathbf{T_\Sigma}$, if each of the $n_b(n_b-1)/2$ elements above the diagonal are smaller or equal to the n_b elements of the diagonal, then the $n_b(n_b-1)/2+1$ lowest value is one of the n_b bubbles. In other words, with n_b bubbles, it is possible to extract at least $n=n_b(n_b-1)/2+1$ values. Reciprocally, by inverting this relation, the minimum value n_b required to extract the n^{th} value will be given by $n_b=\Psi(n)=\text{ceil}((\sqrt{8n+1}-1)/2))$ where ceil(x) represents the smallest integer greater than or equal to x. Consider the example in Fig. 1, where a grey circle represents a candidate that has already been selected to be an element of **E** and a white circle represents a competing bubble for the n^{th} position. In Fig. 1(a), 3 candidates compete for the 5^{th} position in **E,** Fig. 1(b) and 1(c) represent the worst case, or the maximum number of candidates for the n^{th} position for n=7 and 11, respectively. Note that the number of candidates is exactly $\Psi(n)$ when the $(n-1)^{th}$ first elements in **E** form a triangle in $\mathbf{T_\Sigma}$.

Theoretical complexity reduction: As in [3], the core component of the algorithm is the sorter that delivers the lowest value in CB to the output message **E** and replaces it with a new element of $\mathbf{T_\Sigma}$. A corollary of **Theorem 1** is that the maximum theoretical size of the sorter that feeds **E** is $\Psi(n_m)$. This means that the complexity of the elementary check is no longer dominated by $n_{op}\times n_m$ but by $n_{op}\times\Psi(n_m)$, which results in a significant complexity reduction (for example, for n_m=15, the number of comparison operations is reduced by a factor of 3).

The bubble check algorithm: The main originality of the bubble check algorithm is its two-dimensional (2D) strategy for the values in $\mathbf{T_\Sigma}$ that successively feed the sorter. This means that once the element $\mathbf{T}_{\boldsymbol{\Sigma}}(i, j)$ is moved from the sorter to the vector **E**, it can be replaced by either $\mathbf{T}_{\boldsymbol{\Sigma}}(i+1, j)$ or $\mathbf{T}_{\boldsymbol{\Sigma}}(i, j+1)$. To control this, we introduce a flag "horizontal" (H): if H is true, then the value $\mathbf{T}_{\boldsymbol{\Sigma}}(i, j)$ is replaced by $\mathbf{T}_{\boldsymbol{\Sigma}}(i, j+1)$; If H is not true (i.e. Vertical mode), then $\mathbf{T}_{\Sigma}(i, j)$ is replaced by $\mathbf{T}_{\Sigma}(i+1, j)$. The step 2.c) of the EMS algorithm, i.e. the rule to replace the extracted value $\mathbf{T}_{\boldsymbol{\Sigma}}(i, j)$ in the set CB is then modified as follows:

Modified step 2.b)

Flag control: change the value of H.

a) when (i=1) then H=1;

b) when (j=1 and $i \geq n_b$) then H=0.

Test if $\mathbf{T}_{\boldsymbol{\Sigma}}(i+\overline{H}, j+H)$ has already been introduced in the set CB.

If no, then include $\mathbf{T}_{\boldsymbol{\Sigma}}(i+\overline{H}, j+H)$ in CB;

If yes, then include $\mathbf{T}_{\boldsymbol{\Sigma}}(i+H, j+\overline{H})$ in CB.

Fig 2. shows the extraction of the 9 smallest values in $\mathbf{T}_{\Sigma}$ using the bubble check algorithm with n_b=4 for inputs **U**={0,3,5,6,8,…} and **V**={0,2,6,9,10,…}. In this example, the flag changes from horizontal (H=1) to vertical (H=0) at the 7th clock cycle.

Simulation results: We simulated the bubble check with ultra-sparse NB-LDPC codes designed in GF(q=64) and characterized by a fixed variable node degree d_v=2 [4]. Codewords of length N=48, 96, 192, 288, 384 symbols (i.e. a length of Nx6 bits) and rates R=1/2, 2/3, 3/4 and 5/6 were considered. The decoder performs an EMS with horizontal shuffle scheduling and forward/backward processing, with n_m=16, n_{op}=18 and *offset*=1 or 2 (see [3] for the definition of the offset value). Fig. 3 shows simulation results for N=192 and R=1/2, with different number of bubbles in the sorter (n_b = 2, 3, 4, 5 and $\Psi(n_m)$=6). As shown in the figure, there is no performance loss with the bubble check algorithm for $n_b \geq 4$. For $n_b = 3$ and 2, the performance loss is around 0.04 dB and 0.4 dB respectively. The simulation of other code lengths and rates (not shown in this letter) confirms that the performance of the bubble check algorithm with only n_b=4 bubbles remains identical to the performance of the EMS algorithm. This shows that it is possible to further reduce the complexity of the algorithm by using $n_b<\Psi(n_m)$.

Conclusion: In this letter we have presented the bubble check, a novel algorithm for the elementary check node processing of NB-LDPC decoders that introduces significant complexity reduction without performance loss. The check node update being the bottleneck of the decoder complexity, we believe that this complexity reduction is a key feature for possible future hardware implementation (FPGA or ASIC) of NB-LDPC decoders.

Acknowledgement: This work is supported by INFSCO-ICT-216203 DAVINCI "Design And Versatile Implementation of Non-binary wireless Communications based on Innovative LDPC Codes" (www.ict-davinci-codes.eu) funded by the European Commission under the Seventh Framework Program (FP7).

References

1 DECLERCQ, D. and FOSSORIER, M. : “Decoding algorithms for nonbinary LDPC codes over GF(q)”, IEEE Trans. on Communications, vol. 55, pp. 633– 643, April 2007.

2 WYMEERSCH H., STEENDAM H. AND MOENECLAEY M., “Log-Domain Decoding of LDPC Codes over GF(q),” The Proc. IEEE Intern. Conf. on Commun., pp. 772-776, Paris, June 2004.

3 VOICILA A., DECLERCQ D., VERDIER F., FOSSORIER M. and URARD P., “Low-complexity decoding for non-binary LDPC codes in high order fields”, IEEE Trans. on Communications, accepted in 2008.

4 POULLIAT, C., FOSSORIER, M., AND DECLERCQ, D., "Design of regular (2,dc)-LDPC codes over GF(q) using their binary images", IEEE Trans. on Communication, vol. 56(10), pp. 1626 - 1635, Oct. 2008.

Authors’ affiliations:

Université Européenne de Bretagne – UBS, Lab-STICC, Centre de Recherche - BP 92116.

F-56321 Lorient Cedex - FRANCE

Figure captions:

Fig. 1 Candidate values in $\mathbf{T}_\Sigma$ to occupy the n^{th} position of **E.** A grey circle stands for a value already selected from $\mathbf{T}_\Sigma$ and a white circle stands for a candidate value (a bubble).

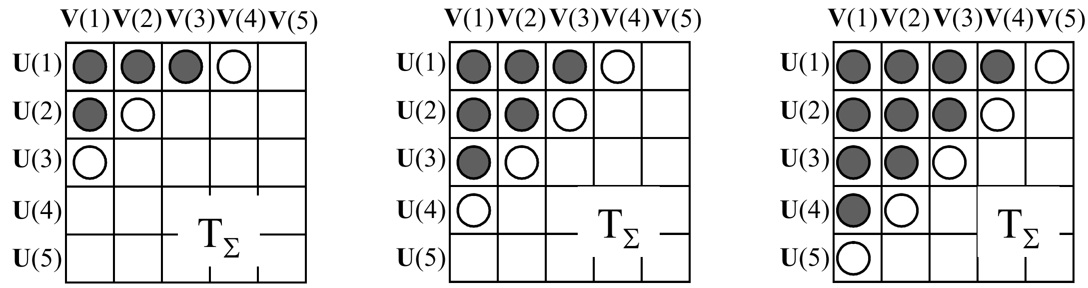
(a) Input vectors **U** and **V,** matrix $\mathbf{T}_\Sigma$ and 3 bubbles for n = 5
(b) Maximum number of candidates for n = 7. The first six values in **E** form a triangle in $\mathbf{T}_\Sigma$
(c) Maximum number of candidates for n = 11. The first ten values in **E** form a triangle in $\mathbf{T}_\Sigma$

Fig. 2 Extraction of the 9 smallest values in $\mathbf{T}_\Sigma$ using the bubble check algorithm with n_b=4 for inputs **U**={0,3,5,6,8,…} and **V**={0,2,6,9,10,…}.

(a) Initial configuration. Extraction of the first value **E**(1)=$\mathbf{T}_\Sigma$(1,1) = 0, flag H = 1 (horizontal mode).
(b) Extraction of the 7th smallest value in $\mathbf{T}_\Sigma$. Since i=n_b, the flag H is flipped to 0 (vertical mode): the next bubble will then be $\mathbf{T}_\Sigma$(5,1).
(c) Extraction of the 9th smallest value of $\mathbf{T}_\Sigma$, H is still equal to 0.

Fig. 3 Simulation results for N=192, R=1/2, n_m = 16, n_{op} = 18, *offset* = 1.0 and at 20 decoding iterations. “EMS” corresponds to the EMS algorithm defined in [3].

Figure 1



a) b) c)

Figure 2

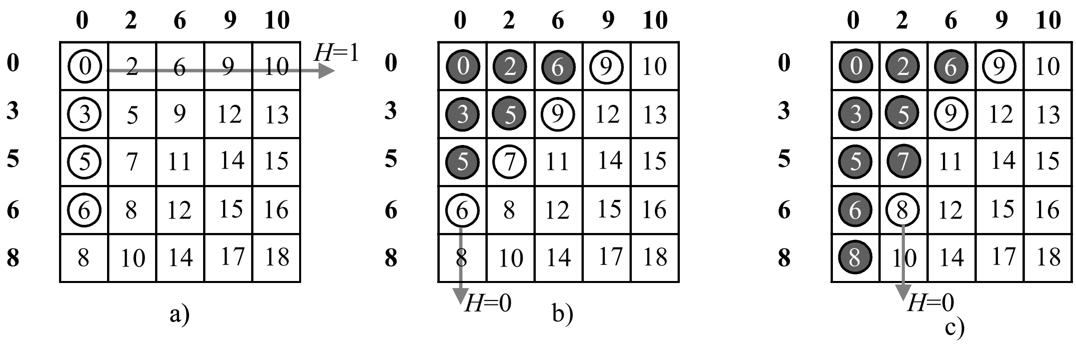
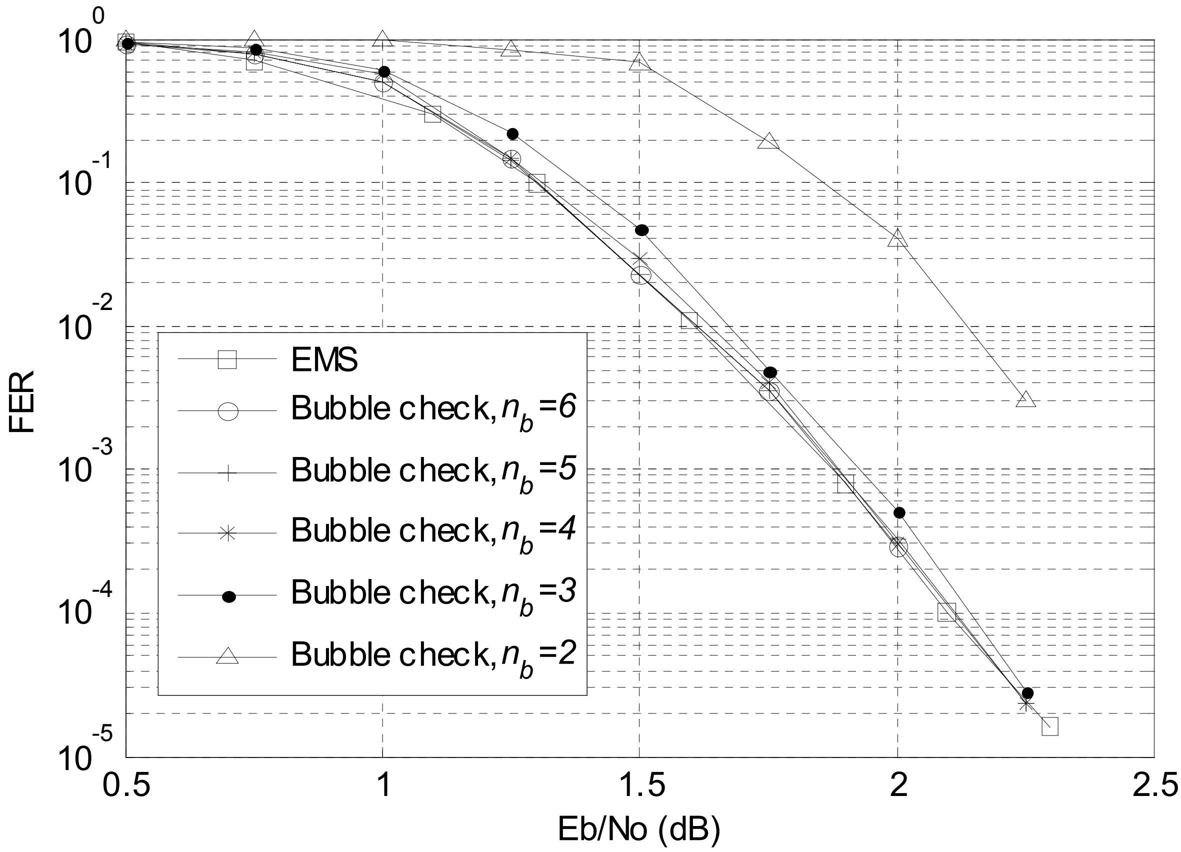


Figure 3



FER
Eb/No (dB)
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
0.5
1
1.5
2
2.5
EMS
Bubble check, n_b=6
Bubble check, n_b=5
Bubble check, n_b=4
Bubble check, n_b=3
Bubble check, n_b=2