Bubble check: a simplified algorithm for elementary check node processing in extended min-sum non-binary LDPC decoders

E. Boutillon and L. Conde-Canencia

A simplified algorithm for the check node processing of extended min-sum non-binary LDPC decoders is proposed. This novel technique, named *bubble check*, can reduce the number of compare operations by a factor of three at the elementary check node level. As this significant complexity reduction is achieved without any performance loss, this technique becomes highly attractive for hardware implementation.

Introduction: Non-binary low density parity-check (NB-LDPC) codes are constructed as a set of parity equations over a Galois field GF(q). Even if they are now known to be an efficient alternative to binary LDPC for the transmission of short frames, their major drawback remains their high decoding complexity, especially at the check node processors. In [1], we highlighted the interest of the extended min-sum (EMS) algorithm applied to NB-LDPC [2, 3] because of the significant complexity reduction it introduces compared to the belief propagation (BP) algorithm. To be specific, the ($q \times \log q$)-BP-complexity is reduced to $n_m \times \log n_m$, where n_m is the size of the truncated probability messages in the decoder ($n_m << q$). However, from a hardware point of view, the EMS complexity is still in the order of n_m^2. In this Letter, we introduce the *bubble check* algorithm, a technique that reduces this complexity to the order of $n_m\sqrt{n_m}$.

EMS elementary check node (ECN) processing: Let us consider a forward/backward implementation of the node update [4]. The check node equation can then be expressed by several elementary steps, defined by a node update that assumes two input messages **U, V** and one output message **E**. These three messages are log-likelihood-ratios (LLR) sorted in increasing order, i.e. $\mathbf{U} = [U(1), U(2), \ldots, U(n_m)]$, with $U(1) = 0$ and $U(i) = -\log(P(U^{gf}(i)/L)/P(U^{gf}(1)/L))$. In this equation, $U^{gf}(i)$ is the GF(q) index associated to $U(i)$ and L are the local hypotheses. Note that the same notation applies for **V** and **E** and that we consider the opposite of the classical LLR definition, which simplifies the global architecture of the decoder [5].

The EMS ECN generates **E**, the output vector containing the n_m smallest values in the set $\{U(i) + V(j)\}$, $(i, j) \in [1, n_m]^2$. This set can be represented as a matrix $\mathbf{T}_\Sigma$, where $\mathbf{T}_\Sigma(i, j) = U(i) + V(j)$. As **U** and **V** are sorted in increasing order, the elements of $\mathbf{T}_\Sigma$ have the following property:

Property 1: $\forall (i, j) \in [1, n_m]^2$, $\forall (i', j') \in [1, n_m]^2$, $i \le i'$ and $j \le j'$ $\Rightarrow$ $\mathbf{T}_\Sigma(i, j) \le \mathbf{T}_\Sigma(i', j')$

EMS ECN algorithm: To obtain the output vector **E**, the authors in [3] use a sorter that contains n_m competing elements from $\mathbf{T}_\Sigma$. This sorter is in fact a subset **B** of the elements of the matrix $\mathbf{T}_\Sigma$, which is initialised with the first column of $\mathbf{T}_\Sigma$ and dynamically updated at each step of the algorithm, as follows:

For $k = 1$ to n_m loop

Step 1: Extract the smallest value in **B**, i.e., $\mathbf{T}_\Sigma(i, j)$. This element becomes a new element of **E**.

Step 2: Replace the extracted value in the sorter by $\mathbf{T}_\Sigma(i, j+1)$.

To extract the smallest value in one clock cycle, the authors in [3] propose the parallel insertion of the new incoming value in **B**. Since this operation is performed n_m times, the global complexity of the EMS is dominated by n_m^2. Note that we omit the explanation of the GF(q) computations that are also performed at the EMS ECN because no novelty is introduced concerning these.

New approach to EMS ECN processing: The principle is to exploit the properties of the values in $\mathbf{T}_\Sigma$ to minimise the size of the sorter and thus reduce the order of complexity of the EMS. From property 1, it follows that $\mathbf{T}_\Sigma(1, 1)$ is the minimum value of $\mathbf{T}_\Sigma$ and it will thus be extracted to occupy the first position of **E** (i.e., $E(1) = \mathbf{T}_\Sigma(1, 1) = 0$). For the second position of **E,** there are two candidates: $\mathbf{T}_\Sigma(2, 1)$ and $\mathbf{T}_\Sigma(1, 2)$. If, for example, $\mathbf{T}_\Sigma(2, 1)$ is extracted (i.e. $E(2) = \mathbf{T}_\Sigma(2, 1) < \mathbf{T}_\Sigma(1, 2)$), then the two candidates for the third position are $\mathbf{T}_\Sigma(3, 1)$ and $\mathbf{T}_\Sigma(1, 2)$, and so on. Note that, for each position, all the candidates belong to a different row and a different column in $\mathbf{T}_\Sigma$.

Fig. 1 presents all the possibilities for the kth position of **E**, with $k = 5$. In this Figure, a grey circle represents a value already extracted from $\mathbf{T}_\Sigma$ to **E** and a white circle represents a candidate (or a *bubble*) for the kth position. The name *bubble check* comes from this graphical representation, i.e. the algorithm uses *bubbles* to go through the elements of matrix $\mathbf{T}_\Sigma$. Let n_b be the number of bubbles for the kth position, in Figs. 1*a*, *c*, *e*, $n_b = 2$ and in Figs. 1*b*, *d*, $n_b = 3$.

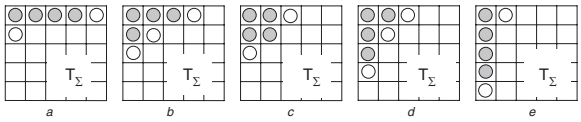

Fig. 1 *Candidate values in $\mathbf{T}_\Sigma$ to occupy fifth position ($k = 5$) of **E***

Grey circle represents element already fed to **E** and white circle represents candidate element (or *bubble*). In *a*, *c* and *e*, $n_b = 2$. In *b* and *d*, $n_b = 3$

At this point, the key question is: what is the maximum number of bubbles needed to perform the algorithm? For each k, the maximum value of n_b corresponds to the worst case, i.e. the already extracted values have a triangular shape in $\mathbf{T}_\Sigma$. If $k - 1$ is a triangular number, (i.e. $k - 1 = t \times (t - 1)/2$, for a given integer t) then the maximum value of n_b is t. Consequently, for all k, n_b can be bounded by the triangular root as:

$$n_b \le \psi(k) = \left\lceil \frac{1 + \sqrt{1 + 8(k-1)}}{2} \right\rceil \tag{2}$$

where $\lceil x \rceil$ represents the smallest integer greater than or equal to x.

Complexity reduction: The maximum theoretical size of the sorter is given by (2) which means that, if we reduce the size of **B** from n_m to $\psi(n_m)$, the complexity of the EMS ECN will be no longer dominated by n_m^2 but by $n_m \times \psi(n_m)$. In a practical implementation of the EMS ECN, for a typical value of $n_m = 15$, $\psi(n_m) = 5$, which means that with this new approach the number of compare operations is reduced by a factor of three.

Bubble check algorithm for EMS ECN: Based on these heuristics, we propose the *bubble check* algorithm, which is novel in that once the element $\mathbf{T}_\Sigma(i, j)$ is moved from **B** to **E**, it can be replaced by either $\mathbf{T}_\Sigma(i+1, j)$ or $\mathbf{T}_\Sigma(i, j+1)$. To implement this, we introduce a 'horizontal' flag, H. If $H = 1$, then $\mathbf{T}_\Sigma(i, j)$ is replaced by $\mathbf{T}_\Sigma(i, j+1)$ in **B**; if $H = 0$, then $\mathbf{T}_\Sigma(i, j)$ is replaced by $\mathbf{T}_\Sigma(i+1, j)$. Step 2 of the EMS algorithm is modified as follows:

Modified step 2: Flag control: change the value of H.

(i) if $(i = 1)$ then $H = 1$, $\bar{H} = 0$
(ii) if $(j = 1$ and $i \ge n_b)$ then $H = 0$, $\bar{H} = 1$
(iii) if $\mathbf{T}_\Sigma(i + H, j + H)$ has never been introduced in **B then** include $\mathbf{T}_\Sigma(i + H, j + H)$ in **B**, else include $\mathbf{T}_\Sigma(i + H, j + \bar{H})$ in **B**

Fig. 2 shows an example of the EMS ECN *bubble check* for inputs $\mathbf{U} = \{0, 7, 15, 21, 25, \ldots\}$ and $\mathbf{V} = \{0, 6, 13, 17, 21, \ldots\}$. The size of the sorter is $n_b = 4$ and the output vector after eight clock cycles is $\mathbf{E} = \{0, 6, 7, 13, 13, 15, 17, 20\}$. The flag changes from $H = 1$ to $H = 0$ at the seventh clock cycle.

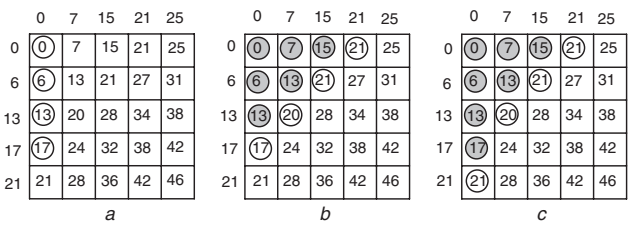

Fig. 2 *Example of ECN processing with bubble check algorithm*

a Initial configuration: $k = 1$, $\mathbf{E}(k) = \mathbf{T}_\Sigma(1, 1) = 0$, $H = 1$
b $k = 7$, $\mathbf{E}(k) = \mathbf{T}_\Sigma(4, 1) = 17$. Since $i = n_b = 4$, $H = 0$, then following bubble in sorter is $\mathbf{T}_\Sigma(5, 1)$
c $k = 8$, $\mathbf{E}(k) = \mathbf{T}_\Sigma(3, 2) = 20$, $H = 0$. Following bubble in sorter is $\mathbf{T}_\Sigma(4, 2)$

Simulation results: We simulated the EMS ECN *bubble check* and the EMS ECN [3] with ultra-sparse NB-LDPC codes designed in GF(64)

and characterised by a fixed variable node degree $d_v = 2$ [6]. We considered horizontal shuffle scheduling, forward/backward processing and $n_m = 16$. Fig. 3 shows simulation results for codewords of length $N = 192$ symbols and rate $R = 1/2$, with $n_b = 2$, 3, 4, 5 and $n_b = \psi(n_m) = 6$. The *bubble check* presents no performance loss for $n_b \geq 4$. For $n_b = 3$ and 2, the performance loss is around 0.04 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 $n_b = 4$ bubbles remains identical to the performance of the EMS algorithm. This shows that, in practice, the complexity can be chosen to be even lower than the theoretical by using $n_b < \psi(n_m)$, without performance loss.

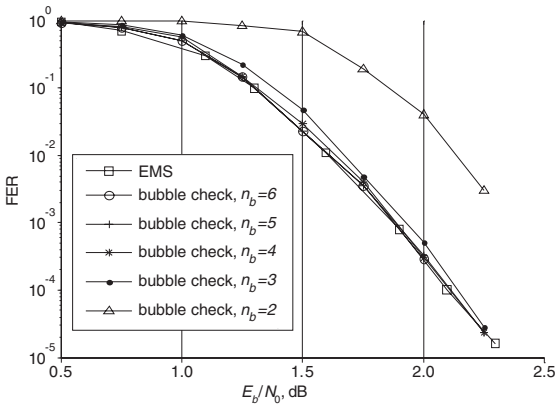

Fig. 3 *Simulation results for $N = 192$, $R = 1/2$, $n_m = 16$ and 20 decoding iterations*

'EMS' corresponds to EMS ECN algorithm defined in [3]. 'Bubble check' is the EMS ECN *bubble check* presented throughout

Conclusion: The *bubble check* is presented as an original algorithm for EMS ECN processing of NB-LDPC decoders. We believe that the $\sqrt{n_m}$-complexity reduction it introduces is a key feature for practical implementation.

Acknowledgments: 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).

10 March 2010
doi: 10.1049/el.2010.0566

E. Boutillon and L. Conde-Canencia (*Université Européenne de Bretagne, UBS. Lab-STICC, CNRS UMR 3192. Centre de Recherche - BP 92116, Lorient Cedex F-56321, France*)

E-mail: laura.conde-canencia@univ-ubs.fr

References

1 Conde-Canencia, L., Al Ghouwayel, A., and Boutillon, E.: 'Complexity comparison of non-binary LDPC decoders'. Proc. ICT Mobile Summit, Santander, Spain, June 2009
2 Declercq, D., and Fossorier, M.: 'Decoding algorithms for nonbinary LDPC codes over GF(q)', *IEEE Trans. Commun.*, 2007, **55**, pp. 633–643
3 Voicila, A., Declercq, D., Verdier, F., Fossorier, M., and Urard, P.: 'Low complexity, low memory EMS algorithm for non-binary LDPC codes'. Proc. of IEEE Int. Conf. on Commun., ICC'2007, Glasgow, United Kingdom, June 2007
4 Wymeersch, H., Steendam, H., and Moeneclaey, M.: 'Log-domain decoding of LDPC codes over GF(q)'. Proc. IEEE Int. Conf. on Commun., ICC'2004, Paris, France, June 2004, pp. 772–776
5 Sevin, V.: 'Min-max decoding for non binary LDPC codes'. Proc. Int. Symp. on Information Theory, ISIT'2008, Toronto, Canada, July 2008, pp. 960–964
6 Poulliat, C., Fossorier, M., and Declercq, D.: 'Design of regular (2,dc)-LDPC codes over GF(q) using their binary images', *IEEE Trans. Commun.*, 2008, **56**, (10), pp. 1626–1635