Low-complexity decoders for non-binary turbo codes

Rami Klaimi, Charbel Abdel Nour, Catherine Douillard, Joumana Farah

Low-complexity decoders for non-binary turbo codes

Rami Klaimi, Charbel Abdel Nour and Catherine Douillard
IMT Atlantique, CNRS UMR 6285 Lab-STICC,
Brest, France
e-mail: firstname.surname@imt-atlantique.fr

Joumana Farah
Faculty of Engineering, Lebanese University,
Roumieh, Lebanon
e-mail: joumanafarah@ul.edu.lb

***Abstract*—Following the increasing interest in non-binary coding schemes, turbo codes over different Galois fields have started to be considered recently. While showing improved performance when compared to their binary counterparts, the decoding complexity of this family of codes remains a main obstacle to their adoption in practical applications. In this work, a new low-complexity variant of the Min-Log-MAP algorithm is proposed. Thanks to the introduction of a bubble sorter for the different metrics used in the Min-Log-MAP decoder, the number of required computations is significantly reduced. A reduction by a factor of 6 in the number of additions and compare-select operations can be achieved with only a minor impact on error rate performance. With the use of an appropriate quantization, the resulting decoder paves the way for a future hardware implementation.**

***Index Terms*—Non-binary turbo codes, decoding algorithm, bubble check, complexity reduction.**

I. Introduction

Non-binary (NB) turbo codes (TC) designed over Galois Fields GF(q) are shown to have a real potential in outperforming existing binary error correction codes [1]. Designed over finite fields and directly mapped to the corresponding Quadrature Amplitude Modulation of the same order (e.g. q-QAM), these codes can benefit from the capacity gain observed between a coded modulation and a bit-interleaved coded modulation [2]. A main obstacle to their wide adoption resides in the fact that their decoding complexity increases with the order q of the finite field. The computational complexity of the symbol-based BCJR algorithm [3] in terms of additions and compare-select (ACS) operations scales as $q^{\nu+1}$, ν being the code memory, whereas the storage requirements for the extrinsic information varies linearly with q and the state metric memory is in the order of q^{ν}.

Complexity-related issues have already been addressed for the implemention of non-binary low-density parity-check (NB-LDPC) codes. Several complexity reduction techniques were proposed in the literature. In [4], [5], the extended Min-Sum (EMS) decoding algorithm was introduced, reducing the computational complexity from the order of q^2 to the order of n_m^2, with $n_m << q$, as shown in [6]. Later, the *bubble check* algorithm was introduced [7], which reduces the computational decoding complexity to the order of $n_m\sqrt{n_m}$.

Inspired by the EMS algorithm simplifications, a modified Min-Log-MAP algorithm is proposed in this work. It targets the reduction of the number of performed ACS operations through the use of a modified bubble check algorithm. The paper is organized as follows: the structure of the considered non-binary component convolutional codes is described in Section II. The proposed low complexity decoding algorithm is presented in Section III and a computational complexity analysis of this algorithm is carried out in Section IV. Comparisons of error correcting performance and complexity results are shown in Section V. Finally, Section VI concludes this work.

II. Non-binary recursive systematic convolutional code sructure

The NB-TC structure considered in this paper is based on the concatenation of two constituent recursive systematic codes designed over GF(q). In order to limit the complexity of the decoder, constituent codes with memory $\nu = 1$ are used, as illustrated in Fig. 1(a). It can be shown that if the code coefficients a_1, a_2 and a_3 are chosen so as to respect $a_1 \neq 0$ and $a_2 + a_3 \neq 0$, the trellis of the resulting code is fully connected and the q^2 transitions in the trellis are labeled by the q^2 possible combinations of the systematic and parity symbols s and p [8]. An example of a fully connected trellis for a code designed on GF(4) is shown in Fig. 1(b).

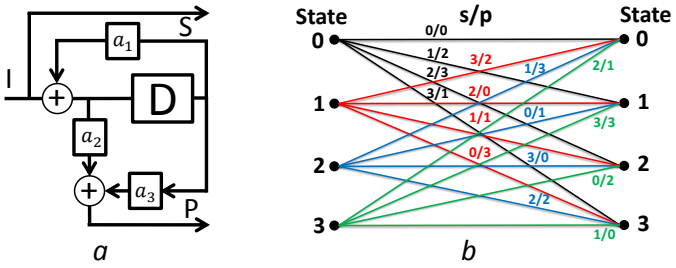

Fig. 1. (a) Structure of a NB recursive convolutional code with one memory element. (b) Trellis of a NB recursive convolutional code defined over GF(4).

III. Simplified decoding of NB convolutional codes

A. Min-Log-MAP decoding of NB convolutional codes

The reference decoding algorithm considered in the study is the scaled Min-Log-MAP algorithm. This algorithm is equivalent to the well-known scaled Max-Log MAP algorithm [9], except that we adopt a log-likelihood ratio (LLR) definition which is the opposite of the conventional definition:

$$L(a) = -\ln \frac{P(x=a)}{P(x=\tilde{a})},\ a \in \mathrm{GF}(q) \tag{1}$$

where $\tilde{a} = \underset{a \in \mathrm{GF}(q)}{\mathrm{Argmax}}\ P(x=a)$.

With this definition, the LLRs are positive and searching for the most likely symbol comes to find the minimum LLR, which is better suited to a compact hardware representation of the LLRs.

Decoding using the Min-Log-MAP algorithm requires the repeated computation of the minimum of cumulated terms (Min-Sum), for the derivation of the state metrics and of the extrinsic LLRs.

The forward state metric related to state $S_i = j$, $j \in \{0 \cdots q-1\}$ at trellis stage i is computed through the forward recursion as:

$$\alpha_i(j) = \min_{j' \in \{0 \cdots q-1\}} (\alpha_{i-1}(j') + \gamma_{s,i-1}(j', j) + \gamma_{p,i-1}(j', j)) \quad (2)$$

where $\alpha_{i-1}(j')$ is the forward state metric related to state $S_{i-1} = j'$ at trellis stage $i-1$, $\gamma_{s,i-1}(j', j)$ and $\gamma_{p,i-1}(j', j)$ are the systematic and parity transition metrics between states $S_{i-1} = j'$ and $S_i = j$, respectively. Similarly, the backward state metric related to state $S_i = j$ at trellis stage i is computed through the backward recursion as:

$$\beta_i(j) = \min_{j' \in \{0 \cdots q-1\}} (\beta_{i+1}(j') + \gamma_{s,i}(j, j') + \gamma_{p,i}(j, j')) \quad (3)$$

Additionally, the extrinsic LLR related to symbol $a \in \mathrm{GF}(q)$ at trellis stage i is computed using:

$$L_i^e(a) = \min_{(j,j') \in \{0 \cdots q-1\}^2 \mid s(j,j')=a} (\alpha_i(j) + \beta_{i+1}(j') + \gamma_{p,i}(j, j')) \quad (4)$$

where $s(j, j')$ is the systematic data symbol in $\mathrm{GF}(q)$ labeling transition (j, j').

The straightforward computation of Eq. 2 or Eq. 3 for the q encoder states requires, at each trellis stage, the computation of q^2 cumulated terms that are compared using $q(q-1)$ compare and select operations. The same holds for the computation of Eq. 4 for the q different symbols in $\mathrm{GF}(q)$. For high values of q, such as 64, 256 or larger, the decoding complexity becomes prohibitive and cannot be efficiently implemented in hardware without simplification. Therefore, we propose a low-complexity algorithm for Min-Log-MAP decoding of NB convolutional codes. The proposal is inspired by the *bubble check* algorithm proposed in [7] to simplify the elementary check node processing of the extended min-sum algorithm for NB-LDPC codes.

B. The bubble check algorithm

The bubble check algorithm described in [7] is based on bubble sorting and aims to simplify check node processing in NB-LDPC decoders using the EMS algorithm. First of all, only a limited number n_m of messages at the input of the check node are considered, in order to reduce the computational burden of the check node update. From two sorted vectors $U = (U(1), U(2), \cdots, U(n_m))$ and $V = (V(1), V(2), \cdots, V(n_m))$, the bubble checker calculates an output vector E, containing the sorted values of the set $\{U(i)+V(j)\}, (i, j) \in [1 \cdots n_m]^2$ in a limited number of operations. An example is illustrated in Fig. 2. The first activated bubble is $(U(1), V(1))$ as indicated in Fig. 2(a). Therefore, $E(1) = U(1)+V(1)$ and then two bubbles can be activated: $(U(1), V(2))$ and $(U(2), V(1))$, as shown in Fig. 2(b). Since $U(1)+V(2) < U(2)+V(1)$, the algorithm sets $E(2) = U(1)+V(2)$ and the bubble $(U(1), V(3))$ is activated next. The process continues until the row or the column number reaches the maximum value allowed for the sorter size. This algorithm was shown to reduce the computational decoding complexity to the order of $n_m\sqrt{n_m}$ [7].

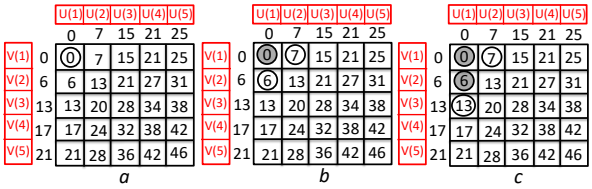

Fig. 2. Example of *bubble check* processing.

However, this algorithm cannot be directly applied for the Min-Sum processing of metrics and LLRs in the Min-Log-MAP algorithm, due to several reasons:

- Min-Sum processing does not require any sorting of the LLRs or cumulated metrics: only their minimum value has to be found;
- the computation of each cumulated term in Eq. 2, Eq. 3 and Eq. 4 involves the addition of three terms instead of two : sorting the cumulated values according to the standard *bubble check* algorithm would require running it twice;
- all the combinations of values are not possible in each sum: in other terms, some cells in the sorting table are empty.

C. Simplified Min-Sum processing for NB convolutional codes

The proposed Min-Sum processing algorithm takes these differences into consideration. Due to the second point, two sorting processes should be used in turn, requiring two sorting tables. However, in order to limit the computational complexity, we implemented the search for the minimum cumulated term using only one table, with rows and columns indexed with two among the three terms of the sum.

The Min-Log-MAP algorithm requires, at every trellis stage and in each iteration, q^2 addition and comparison operations for its recursive calculation of the forward and backward state metrics. Inspired by the algorithm described in [7], a new low complexity decoder is proposed next. In the following, we detail the application of the proposed algorithm for the computation of the forward state metrics. However, the same algorithm can be applied for the backward metrics and the extrinsic LLRs.

Considering the calculation of the forward metric $\alpha_i(j)$ given by Eq. 2, the sorting table contains the q values of the different terms $B(j', j) = \alpha_{i-1}(j') + \gamma_{s,i-1}(j', j) + \gamma_{p,i-1}(j', j)$, called bubbles, by analogy with [7]. The rows and columns of the table are arranged according to the values of the forward metrics $\alpha_{i-1}(j')$ and the values of the systematic transition metrics $\gamma_{s,i-1}(j', j)$, both sorted in increasing order: $\{\alpha_{k_\alpha}\}, k_\alpha = \{1 \cdots q\}$ and $\{\gamma_{s,k_s}\}, k_s = \{1 \cdots q\}$. Each bubble value $B(j', j)$ is placed in the table at the intersection of the

corresponding $\alpha_{i-1}(j')$ and $\gamma_{s,i-1}(j',j)$ values. The parity transition metrics $\gamma_{p,i-1}(j',j)$ are sorted separately in increasing order following index $k_p = \{1 \cdots q\}$.

In the fully connected trellis of the code structure described in Fig. 1, each transition corresponds to one of the q^2 cells of the table. However, when computing Eq. 2, only the q transitions merging at state $S_i = j$ are considered. In the studied code structure, all the transitions merging at a given state are labeled with different values of systematic symbols. Therefore, the q values of $B(j',j)$ in the table are all placed in different rows and columns. An example of such a table is shown in Fig. 3 for $q = 8$.

To reduce the number of computations involved in Min-Sum processing, a preliminary step consists in defining a radius R which sets the boundaries of the table region corresponding to the R^2 lowest, i.e. most reliable, values of $\alpha_{i-1}(j') + \gamma_{s,i-1}(j',j)$. Before starting the computation, the proposed algorithm checks whether this high reliability zone (colored in blue in Fig. 3) contains at least one bubble. If not, it is highly unlikely that the considered state has been visited by the encoder. Therefore, the corresponding $\alpha_i(j)$ value is set to a predefined high value. The value of R has an impact on the complexity and the error rate performance of the decoder, as shown in Section V.

If the radius-R region contains one bubble or more, the computation process can be launched according to the flowchart of Fig. 4. The algorithm processes the bubbles alternately vertically and horizontally (or vice-versa) and updates two upper bounds, $k_{\alpha,\max}$ and $k_{s,\max}$, on the maximum values of indexes k_α and k_s to be processed.

- ***Step 1 – Initialization:*** Initialize k_α, k_s and k_p to 1 and initialize $k_{\alpha,\max}$, $k_{s,\max}$ to q.
- ***Step 2 – Vertical processing:*** 1) Calculate the bubble in the column indexed by k_α. 2) Update $\alpha_i(j)$ with the computed bubble $B_i(j',j)$ if $B_i(j',j) < \alpha_i(j)$. 3) If the parity transition metric term $\gamma_{p,i-1}(j',j)$ in $B_i(j',j)$ is γ_{p,k_p}, increment k_p.
- ***Step 3 – Update $k_{s,\max}$ and increment k_α:*** Let m be the row number of the systematic transition metric term $\gamma_{s,i-1}(j',j)$ used to compute $B_i(j',j)$ in step 2. Compute a dummy bubble B' with the lowest values of $\alpha_{i-1}(j')$ and $\gamma_{p,i-1}(j',j)$ not yet used (that is with indexes $k_\alpha + 1$ and k_p due to the applied sorting) and with $\gamma_{s,m+1}$: $B' = \alpha_{k_\alpha+1} + \gamma_{s,m+1} + \gamma_{p,k_p}$. This dummy bubble sets a lower bound on the values of the actual bubbles located at rows with indexes greater than $m+1$. If $B' \geq \alpha_i(j)$, there is no need to continue the calculation process below this row: $k_{s,\max}$ is set to $m+1$. k_α is incremented for the next iteration.
- ***Step 4 – Horizontal processing:*** 1) Calculate the bubble in the row indexed by k_s. 2) Update $\alpha_i(j)$ with the computed bubble $B_i(j',j)$ if $B_i(j',j) < \alpha_i(j)$. 3) If the parity transition metric term $\gamma_{p,i-1}(j',j)$ in $B_i(j',j)$ is γ_{p,k_p}, increment k_p.
- ***Step 5 – Update $k_{\alpha,\max}$ and increment k_s:*** Let n be the column number of the state metric term $\alpha_{i-1}(j')$ used to compute $B_i(j',j)$ in step 4. Compute a dummy bubble B'' with the lowest values of $\gamma_{s,i-1}(j',j)$ and $\gamma_{p,i-1}(j',j)$ not yet used (that is with indexes $k_s + 1$ and k_p due to the applied sorting) and with α_{n+1}: $B'' = \alpha_{n+1} + \gamma_{s,k_s+1} + \gamma_{p,k_p}$. This dummy bubble sets a lower bound on the values of the actual bubbles located at columns with indexes greater than $n+1$. If $B'' \geq \alpha_i(j)$, there is no need to continue the calculation process to the right of this column: $k_{\alpha,\max}$ is set to $n+1$. k_s is incremented for the next iteration.

Steps 2 to 5 are repeated until $k_s \geq k_{s,\max}$ and $k_\alpha \geq k_{\alpha,\max}$.

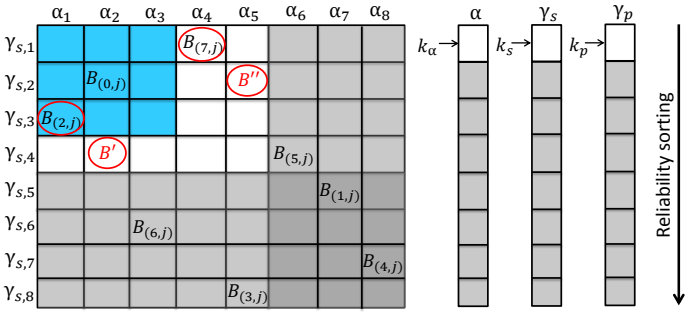

Fig. 3. Example of sorting table used for the computation of $\alpha_i(j)$ with Eq. 2 for $q = 8$. All $B(j',j)$ values, $j = 0 \cdots q-1$ are placed in different rows and columns. Min-Sum processing is performed with radius $R = 3$, $k_{s,\max} = 5$ and $k_{\alpha,\max} = 6$. Illustration of Steps 2 and 3 (resp. Steps 4 and 5) for $k_\alpha = 1$ (resp. for $k_s = 1$), and illustration of sorted vectors α, γ_s and γ_p.

The proposed decoding algorithm limits the activated bubbles to the ones lying inside the area bounded by $k_{\alpha,\max}$ and $k_{s,\max}$.

The same process is repeated for every state $j \in \{0 \cdots q-1\}$ and at each trellis stage.

The proposed algorithm can also be directly applied for the computation of backward state metrics and extrinsic LLRs.

A question that arises is the choice of the metric to be handled separately in the process (e.g the parity transition metric in the example above). Although any metric could play this role, in practice it is better to keep in the table the metrics that are refined during the iterative decoding process thanks to the incoming extrinsic information, i.e. the state metric and the systematic transition metric: this speeds up the computation over the iterations.

D. Additional simplification and vector sorting

In order to further reduce the complexity of the decoding process, the previously described algorithm does not need to consider all the q state metrics and systematic transition metrics to find the minimum bubble value. Similarly to the EMS decoding algorithm of NB-LDPC codes, only the first n_m lowest (i.e. most reliable) metrics need to be sorted, with $n_m << q$. Therefore, in the algorithm of Section III-C, q can be replaced by n_m. The impact of the value of n_m on the complexity and error correction performance is assessed in Section V.

An example of hardware architecture for the generation of the sorted truncated vectors can be found in [10].

IV. Complexity analysis

The computational complexity for the calculation of the forward state metrics, backward state metrics and extrinsic

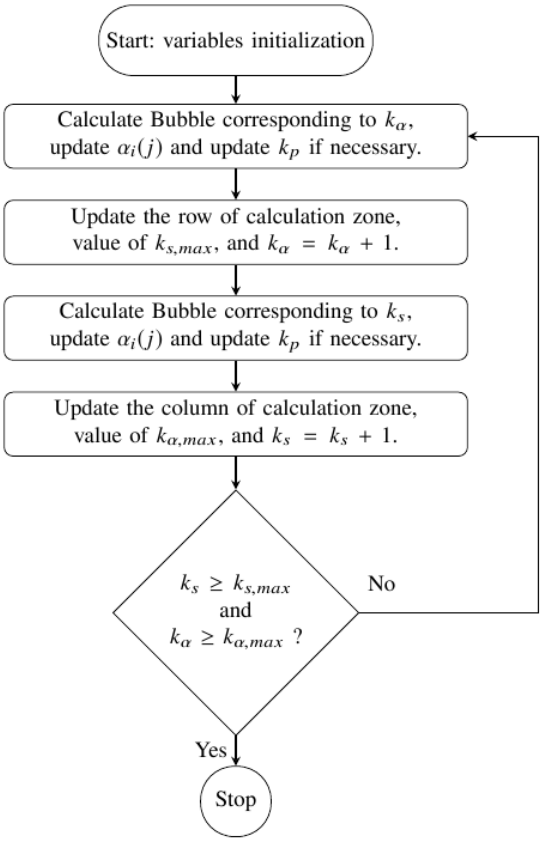

Fig. 4. Flow chart of the proposed simplified Min-Sum processing algorithm.

LLRs is assessed in terms of the average number of ACS operations executed per decoded frame. For the full Min-Log-MAP algorithm, it is calculated as follows:

$$ACS_{MLM} = 3n_{enc}.K_s.n_{it}.q^2 \quad (5)$$

where n_{enc} denotes the number of component encoders in the TC structure – for conventional TCs, $n_{enc} = 2$, K_s is the number of GF(q) symbols in the messages to be encoded and n_{it} represents the number of executed decoded iterations. The factor 3 stands for the three processes needed to compute the forward and backward probabilities and the extrinsic information.

For the proposed bubble check algorithm, the number of operations ACS_{BC} is upper bounded by:

$$ACS_{BC} = 3n_{enc}.K_s.n_{it}.2n_m.q \quad (6)$$

The factor 2 in (6) accounts for the update of the metrics in Steps 2 or 4 of the algorithm (see Section III-C) and of the update of $k_{\max}$ in Steps 3 or 5: for each of the q encoder states, at most $2n_m$ ACS operations are needed. Therefore, from (5) and (6), it can be inferred that the proposed Min-Sum processing reduces the complexity of the Min-Log-MAP algorithm, provided that $n_m < \frac{q}{2}$.

V. Results and discussion

The error correction performance of the proposed decoder in terms of frame error rate (FER) is evaluated by simulations over a Gaussian channel. The simulated NB-TC consists of two identical constituent codes with the structure shown in Fig. 1 and defined over GF(64). The coefficients a_1, a_2 and a_3 are taken equal to 41, 2 and 0, respectively. Since no puncturing is applied, the overall coding rate is $R = 1/3$. The message length at the encoder input is $K_s = 900$ symbols, corresponding to $K_b = 5400$ bits. The coded symbols are transmitted using a 64-QAM constellation. An almost regular permutation (ARP) [11] is used for internal interleaving. Interleaver parameters and corresponding minimum spread S_{min} and girth values are given in Table I.

The turbo decoding algorithms are simulated using $n_{it} = 8$ decoding iterations, unless otherwise specified. 8 quantization bits are used for the representation of the input symbol LLRs and 9 bits for the state metrics – forward and backward.

Three decoding configurations, shown in Table II, are compared in terms of error rate performance and complexity, which differ in the radius value R and in the truncation length n_m (see Section III-D).

TABLE I
ARP interleaver parameters, GF(64), $K_s = 900$ symbols and $R = 1/3$ (ARP equation: $\Pi(i) = (Pi + S(i \bmod Q)) \bmod K_s$).

S_{min}	Girth	P	Q	(S(0),...,S(Q-1))
30	8	137	4	(0,854,396,362)

TABLE II
Values of parameters R and n_m for the three simulated configurations.

Configuration	C1	C2	C3
R	10	4	2
n_m	16	8	4

Fig. 5 shows the error correction loss due to the reduced-complexity decoding algorithm, with respect to the original Min-Log-MAP algorithm. This loss varies from 0.1 to 0.9 dB at a FER equal to 10^{-3}, depending on the configuration. Two additional curves were added, corresponding to the simulation of two binary TCs using 16-state and 64-state constituent codes and decoded with the Min-Log-MAP algorithm under the same simulation conditions. We can observe that the proposed NB-TC outperforms both binary codes even when a complexity reduction is applied. The proposed code can compete with the binary TC using the same number of states, even when performing only 6 decoding iterations (curve labeled C3-R2-n_m4-6It) and even with the binary 16-state binary turbo code, which displays a lower decoding threshold than the binary 64-state code.

A comparison of the computational complexity was also carried out. First, Table III presents the values of the upper bound on the number of ACS for the proposed reduced-complexity decoding algorithm obtained according to (6) and that of the full Min-Log-MAP algorithm obtained with (5), as well as the percentage gain for the three simulated configurations.

Fig. 6 compares the actual computational complexity of the different schemes in terms of measured ACS operations as a function of the FER. The complexity reduction with respect to the classical scaled Min-Log-MAP algorithm varies from

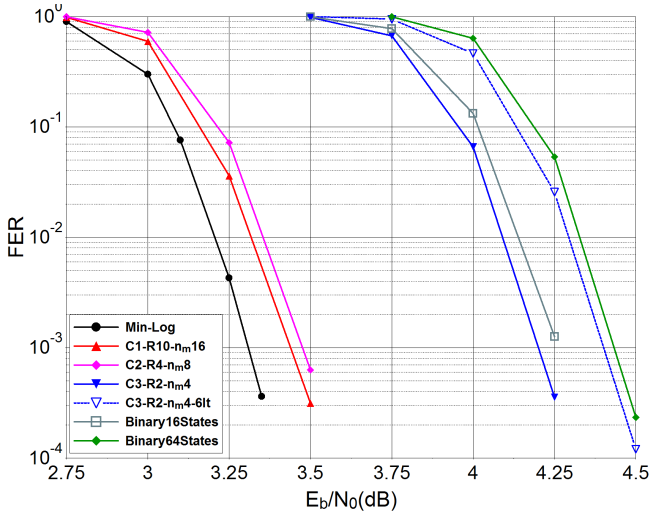

Fig. 5. Performance comparison, in terms of FER, of the 3 low-complexity decoding configurations with Min-Log-MAP algorithm for NB-TC, and 16-state and 64-state binary TC. Transmission using 64-QAM over the AWGN channel. Generator polynomial for binary 16-state and 64-state component codes: $(1, 35/23)$ and $(1, 171/133)$ respectively, in octal notation.

TABLE III
UPPER BOUNDS ON THE NUMBER OF ACS OPERATIONS BY FRAME FOR DIFFERENT DECODING ALGORITHMS.

Algorithm	MLM	Bubble Check		
		$n_m = 16$	$n_m = 8$	$n_m = 4$
# ACS×10^6/Frame	176.9	88.4	44.2	22.1
Min. complexity reduction (%)	0	50.1	75.1	87.5

a factor 3 to almost 10, depending on the configuration. For instance, accepting a loss in the order of 0.2 dB in performance will result in a complexity reduction by a factor of 6 using configuration $C2$. When comparing with the binary 16-state, the NB-TC decoded with the Min-Log-MAP algorithm is 20 times more complex and 6 times when comparing with the 64-state TCs, for a corresponding gain in performance of 0.9 dB and 1.2 dB. With the simplified configuration $C2$, a performance gain of 0.8 dB can still be achieved compared to the 16-state TC, at the price of a complexity multiplied by 4. Note that, although not shown in the figure, we observed that the NB-TC is able to achieve lower error floors than the 16-state binary TC.

Moreover, even when requiring a higher number of operations, the decoder of a NB-TC over GF(q) would enhance the throughput, compared to the binary TC, thanks to symbol-based processing that provides $\log_2(q)$ bits when decoding a trellis section. Indeed, comparisons should be performed at the same throughput level which would require considering high order radix-based decoder for binary codes.

VI. Conclusion

In this work, a reduced-complexity decoding algorithm for NB-TC is proposed. It extends the bubble check algorithm used for NB-LDPC codes to the particular case of metric computations of NB turbo decoders. Based on the Min-Log-MAP decoder, this algorithm largely reduces the number of

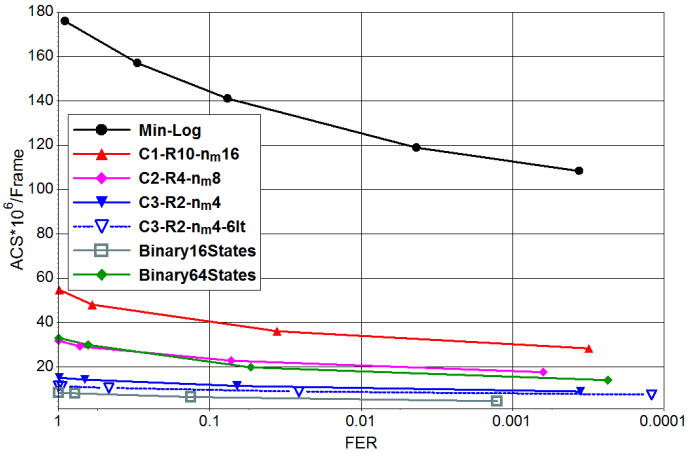

Fig. 6. Computational complexity comparison in terms of ACS operations as a function of the achieved FER.

required addition/comparison operations. Different trade-off levels can be achieved between performance and complexity.

When compared to their binary counterparts, the NB-TCs still show better performance at an affordable additional complexity. This paves the way to future hardware implementations.

Acknowledgment

This work was partially funded by the EPIC project of the EU's Horizon 2020 research and innovation programme under grant agreement No. 760150, by Orange Labs and by the Pracom cluster. It has also received support from the PHC CEDRE program.

References

[1] G. Liva, E. Paolini, B. Matuz, S. Scalise, and M. Chiani, "Short turbo codes over high order fields," *IEEE Trans. Commun.*, vol. 61, no. 6, pp. 2201–2211, June 2013.

[2] G. Caire, G. Taricco, and E. Biglieri, "Bit-interleaved coded modulation," *IEEE Trans. Inform. Theory*, vol. 44, no. 3, pp. 927–946, 1998.

[3] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate (corresp.)," *IEEE Trans. Inform. Theory*, vol. 20, no. 2, pp. 284–287, 1974.

[4] D. Declercq and M. Fossorier, "Decoding algorithms for nonbinary LDPC codes over GF (q)," *IEEE Trans. Commun.*, vol. 55, no. 4, pp. 633–643, 2007.

[5] A. Voicila, D. Declercq, F. Verdier, M. Fossorier, and P. Urard, "Low-complexity decoding for non-binary LDPC codes in high order fields," *IEEE Trans. Commun.*, vol. 58, no. 5, pp. 1365–1375, 2010.

[6] L. Conde-Canencia, A. Al-Ghouwayel, and E. Boutillon, "Complexity comparison of non-binary LDPC decoders," in *ICT MobileSummit*, 2009, pp. 1–8.

[7] E. Boutillon and L. Conde-Canencia, "Bubble check: a simplified algorithm for elementary check node processing in extended min-sum non-binary LDPC decoders," *Electron. Lett.*, vol. 46, no. 9, pp. 633–634, 2010.

[8] R. Klaimi, C. Abdel Nour, C. Douillard, and J. Farah, "Design of low-complexity convolutional codes over GF (q)," *arXiv preprint arXiv:1807.02481*, 2018.

[9] J. Vogt and A. Finger, "Improving the Max-Log-MAP turbo decoder," *Electron. Lett.*, vol. 36, no. 23, pp. 1937–1939, Nov 2000.

[10] A. A. Ghouwayel and E. Boutillon, "A systolic LLR generation architecture for non-binary LDPC decoders," *IEEE Commun. Lett.*, vol. 15, no. 8, pp. 851–853, August 2011.

[11] C. Berrou, Y. Saouter, C. Douillard, S. Kerouédan, and M. Jézéquel, "Designing good permutations for turbo codes: towards a single model," in *IEEE Int. Conf. Communications*, vol. 1, Paris, France, 2004, pp. 341–345.