Low Complexity Non-binary Turbo Decoding based on the Local-SOVA Algorithm

Hugo Le Blevec, Rami Klaimi, Stefan Weithoffer, Charbel Abdel Nour, Amer Baghdadi

Low Complexity Non-binary Turbo Decoding based on the Local-SOVA Algorithm

Hugo Le Blevec, Rami Klaimi, Stefan Weithoffer, Charbel Abdel Nour, Amer Baghdadi
Email: firstname.lastname@imt-atlantique.fr
IMT Atlantique, Lab-STICC, UMR CNRS 6285, F-29238 Brest, France

***Abstract*—Non-binary Turbo codes have been shown to outperform their binary counterparts in terms of error correcting performance yet the decoding complexity of the commonly used Min-Log-MAP algorithm prohibits efficient hardware implementations. In this work, we apply for the first time the recently proposed Local SOVA algorithm for decoding non-binary Turbo codes. Moreover, we propose a low complexity variant dedicated to the direct association with high order constellations denoted by the nearest neighbor Local SOVA. It considers only a limited amount of nearest competing constellation symbols for the soft output computation. Simulation results show that this approach allows a complexity reduction of up to 52% in terms of add-compare-select operations while maintaining the same error correcting performance compared to the Min-Log-MAP algorithm. It can even reach up to 80% if high code rates or frame error rates higher than 10^{-4} are targeted. The achieved complexity reduction represents a significant step forward towards hardware implementation.**

***Keywords*—Non-binary Turbo codes, low complexity decoding, Local SOVA**

I. Introduction

Since their invention, binary turbo codes [1] and iterative decoding have found their way into a wide range of communication standards like the third and fourth generations (3G, 4G) of mobile communications. Furthermore, they have participated to the rediscovery of low density parity check (LDPC) codes [2]. Benefiting from a direct mapping to high order constellation symbols [3], the non-binary (NB) variants for both code classes defined over Galois Fields GF(q) have been shown to outperform their binary counterparts. However, the corresponding high decoding complexity imposes limits on the Galois Field order q and on the achievable frame sizes in practice [4], [5]. For non-binary low density parity check (NB-LDPC) codes, these complexity issues have been addressed in a number of works [6], [7] paving the way for hardware implementations. On the other hand, the strong focus on the *Min-Log-MAP* (MLM) algorithm, which has long been the standard decoding algorithm for binary Turbo codes, has carried over to a large extent to NB Turbo decoding. In [8], complexity reduced MLM decoding was proposed for NB Turbo decoding, inspired by the *bubble check* algorithm (BC) [7]. To calculate the state metric recursion and the extrinsic information, this algorithm resorts to a subset of the $n_m < q$ highly reliable transitions in the code trellis. These n_m transitions are processed using a 2D-*bubble sorter* for the *add compare select* (ACS) operations. A complexity reduction ranging from 50% to 75% in terms of ACS operations was achieved at the cost of < 0.3dB in terms of *frame error rate* (FER) compared to the MLM.

Recently, the *Local-SOVA* (LSOVA) algorithm was proposed as a reduced-complexity alternative to the MLM for binary Turbo codes [9]. Based on a path-centric view on the code trellis, LSOVA was shown to achieve a computational complexity reduction of up to 37% compared to the MLM with a penalty of less than 0.1dB in terms of FER for radix-8 decoding. Practical implementation in 22nm technology [10] exceeded computational complexity predictions to achieve around 45% of reduction.

In this work, we apply for the first time the LSOVA algorithm to the decoding of NB-Turbo codes and propose additional complexity reductions dedicated to the NB case. We achieve a reduction of up to 80% compared to the MLM in terms of ACS operations, getting us one step forward towards efficient hardware implementations for NB Turbo decoding.

The remainder of this paper is structured as follows: Section II gives the necessary background on NB-Turbo decoding and Section III formulates the LSOVA for the decoding of NB-Turbo codes (NB-TC). The proposed method to reduce the complexity of the LSOVA is presented in Section IV, while its corresponding complexity is addressed in Section V. To assess the impact of the simplifications on the error correcting performance, simulation results for different code rates are shown in Section VI. Comparisons with state-of-the-art are performed in Section VII. Section VIII concludes the paper.

II. Non-Binary Turbo Decoding

1) Non-Binary Turbo Codes – Structure and Constituent Codes: In the following, we consider a NB-TC consisting of two concatenated recursive systematic convolutional codes over GF(q). These latter follow the structure illustrated in Fig. 1-a. Composed of one memory element, it provides an excellent tradeoff between complexity and error correcting performance [8]. The code coefficients a_1, a_2 and a_3 respect $a_1 \neq 0$ and $a_2 + a_3 \neq 0$ leading to a fully connected trellis [4]. In other words, the q^2 transitions in the trellis are labeled by the q^2 possible combinations of systematic s and parity p symbols as shown in Fig. 1-b.

2) Min-Log-MAP decoding of NB convolutional codes: Standard component decoding algorithm for binary Turbo codes is the MLM algorithm [11] and its symbol based variant is considered here as reference [12]. For a clear link to related work, we will adopt in the following the notation from [9].

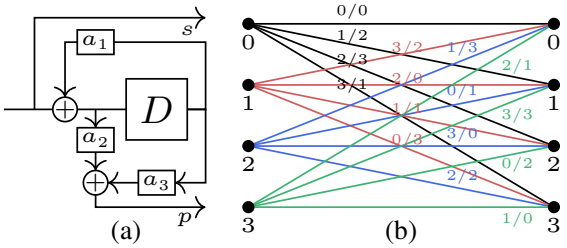

Figure 1: (a) Structure of the NB recursive convolutional component code considered in this work. (b) Trellis of a NB recursive convolutional code over GF(4).

The *symbol-based* MLM algorithm provides for each symbol $d = i$, with $i \in \{0, ..., q-1\}$ and trellis step k, an estimate on the *a Posteriori Logarithmic Probability* (ALP) L_k^i:

$$L_k^i = \min_{(s,s')|d_k=i} \left(A_k(s) + \Gamma_k(s', s) + B_{k+1}(s')\right) \quad (1)$$

In Eq. (1), (s, s') refers to the transition from state s to state s' and $\Gamma_k(s, s')$ to the associated branch in trellis step k. $A_k(s)$ and $B_{k+1}(s')$ are the *forward-* and *backward state metrics* for states $s, s' \in \{0, ..., q-1\}$ and are calculated recursively by

$$A_k(s) = \min_{(s,s')} \left(A_{k-1}(s') + \Gamma_{k-1}(s, s')\right) \quad (2)$$

$$B_k(s') = \min_{(s,s')} \left(B_{k+1}(s) + \Gamma_k(s, s')\right) \quad (3)$$

The complexity of computing Eqs. (1), (2) and (3) is dictated by the galois field order q, since for the fully connected trellis q^2 terms (for the q^2 transitions (s, s')) have to be compared.

III. The Local SOVA algorithm for NB decoding

The LSOVA algorithm was proposed in [9] and has since been demonstrated to allow reduced complexity hardware implementation [10]. In contrast to the MLM, which operates on branches (s, s') in the trellis, the LSOVA operates on *Paths* along r trellis steps $k, k+1, ..., k+(r-1)$ for a radix order of $R = 2^r$. A path P is defined as

$$P = \{M,\ u,\ L\} \in \mathbb{R} \times \{0,\ q-1\}^r \times \{\mathbb{R}^+\}^r \quad (4)$$

where M is the path metric computed through $M = A_k(s) + \Gamma_k(s, s') + B_{k+(r-1)}(s')$, u is the hard decision estimate corresponding to the set of symbols d_i along the r trellis steps of the path, and L corresponds to the set of associated reliability values. Fig. 2 illustrates the path definition for two

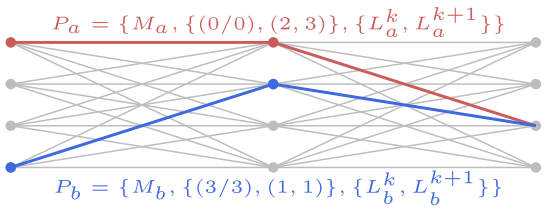

Figure 2: Two radix-4 paths in the trellis of Fig. 1 (b).

radix-4 paths in the trellis of Fig. 1 (b). In the following, we assume $r = 1$, so that a path corresponds to a single branch in the trellis. Based on the path definition from Eq. (4), the LSOVA requires a *Merge* operation to consolidate two paths $P_a = \{M_a, u^a, L^a\}$ and $P_b = \{M_b, u^b, L^b\}$ into one path $P_c = \{M_c, u^c, L^c\}$. Therefore, it has three functions to update M, u and L:

$$M_c = f_0(M_a, M_b) = \min(M_a, M_b) \quad (5)$$

$$u^c = f_1(u^a, u^b) = \begin{cases} u^a & \text{, if } f_0(M_a, M_b) = M_a \\ u^b & \text{, if } f_0(M_a, M_b) = M_b \end{cases} \quad (6)$$

$$L^c = \left\{L^c | L^c = f_2(L^a, L^b)\right\}. \quad (7)$$

Where f_2 corresponds to the Hagenauer rule (HR) [13] or the Battail rule (BR) [14], [15]:

$$L^c = f_2(L^a, L^b) \quad (8)$$
$$= \begin{cases} \min\left(L_{p'}, \Delta_{p,p'}\right) & \text{, if } u^a \neq u^b \text{ (HR)} \\ \min\left(L_{p'}, \Delta_{p,p'} + L_p\right) & \text{, if } u^a = u^b \text{ (BR).} \end{cases}$$

BR is used in the case of identical symbols $u^a = u^b$, whereas for the case of differing symbols $u^a \neq u^b$, HR is used. In Eq. (8), $p' = \text{argmin}_{a,b}(M_a, M_b)$, $p = \text{argmax}_{a,b}(M_a, M_b)$ and $\Delta_{p,p'} = M_p - M_{p'}$.

When merging all paths at a trellis stage k into P_c, this latter can be considered as the Maximum Likelihood (ML) path for the symbol. It was proven in [9] that the merge operation is associative and commutative, which allows for an arbitrary ordering of the necessary merge operations. On the one hand, applying an ordering that merges paths labeled by the same information symbols d_k leads to the MLM [9]. On the other hand, applying a two-phase ordering where, in a first phase (in [10] called *ACS phase*), paths with identical ending state s are merged allows to avoid BR updates entirely for radix-2 (i.e. $r = 1$) decoding, since branches leading to the same state are always labeled by different symbols. In a second phase (*SOU phase*) the resulting winning paths are then merged together, minimizing the overall amount of needed BR updates.

IV. The nearest-neighbors approach

In the following, we assume the GF(q) symbols to be mapped to constellation symbols directly as in [4]. In the constellation diagram (see example for GF(64) in Fig. 3), the minimum distance d_0 between two points in the constellation is well defined as $d_0 = (2/\bar{E}_s)^2$ where $\bar{E}_s$ is the average energy per symbol. Over an Additive White Gaussian Noise (AWGN) channel, from the Union Bound (UB) [1], the probability of error decays exponentially with the coding gain jointly composed of the code rate and the accumulated Euclidean distance along the diverging-converging (DC) trellis sequences [16]. Therefore, the constellation points located close to the received symbol have an exponentially larger likelihood of being the transmitted symbol compared to the far ones. Consequently, for a considered symbol s we propose to consider only a subset of paths within a radius R_s that encompasses a certain number of neighboring candidate symbols for which the update rules of Eq. (8) are applied. These paths

are used for the update of the L-values using the distance between constellation symbols as a preliminary information about their reliabilities:

$$R_s = s \cdot d_0, 1 \leq s \leq \log_2(q). \tag{9}$$

L_i is updated if the symbol i verifies $D(u,i) \leq R_s$ where u is the symbol carried by the path with the highest metric M and D is the Euclidean distance. By applying this method, the number of operations required during each merge is limited. Hence, considering the total amount of performed merges, this method can considerably reduce the overall complexity of the decoding. Note that for a certain radius R_s, the number of points considered will differ with the position of the symbol in the constellation: symbols located close to the edges will contain less points within R_s than points in the middle. Therefore, the achievable complexity reduction is expected to vary depending on the input frame. We refer to the proposed approach as *nearest neighbor approach* (NN-LSOVA) and will illustrate it with the following example.

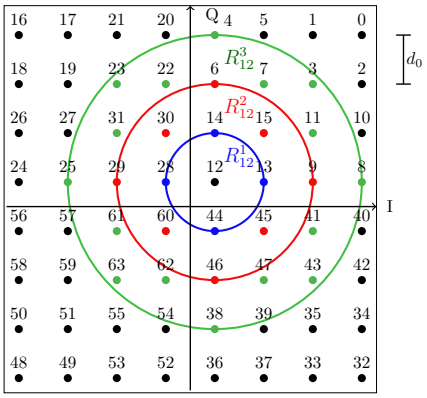

Figure 3: 64-QAM with some R_s configurations.

Consider the case where $q = 64$ and let P_1 and P_2 be two paths such that $P_1 = \{M_1, u_1, L_1\}$ and $P_2 = \{M_2, u_2, L_2\}$ where $L_1 = \{L_1^0, L_1^1, ..., L_1^{63}\}$ and $L_2 = \{L_2^0, L_2^1, ..., L_2^{63}\}$. Let's assume that $u_1 = 12 \neq u_2$, $M_1 \geq M_2$ and the considered radius is $R_1 = d_0 = 1$. When merging the two paths with the LSOVA algorithm, the update rule is applied to all the L-values, resulting in 64 update operations. However, the NN-LSOVA will only apply the update rule on the L-values corresponding to symbols in the considered radius. Here symbols 12, 13, 14, 28 and 44 are to be updated, which means only computing $f_2(L_1^{12}, L_2^{12})$, $f_2(L_1^{13}, L_2^{13})$, $f_2(L_1^{14}, L_2^{14})$, $f_2(L_1^{28}, L_2^{28})$ and $f_2(L_1^{44}, L_2^{44})$. For all the other symbols, the values carried by the winning path P_1 will be kept, lowering the amount of update operations to only five.

The NN-LSOVA offers then a new trade-off between complexity and performance that we will discuss hereafter.

V. Complexity of the NN-LSOVA

In the following, we call layer a set of 2-by-2 merge operations executed in parallel. The merging of paths in the SOU phase is carried out in the LSOVA algorithm following a binary 2-by-2 tree expressed as a sequence of layers. We denote by $N_{l,q}$ the amount of 2 by 2 merge operations to be performed in a layer l. This number depends on the number of symbols q and the index of the layer l and can be computed as $N_{l,q} = \frac{q}{2^l}$. For a non-binary code over GF(q), there are $\log_2(q)$ layers: the first one having q paths, the second $\frac{q}{2}$, the third $\frac{q}{4}$ until choosing the winning path. Further, we evaluate the amount of L-value update operations for layer l considering neighbours within R_s^l, by $\lambda_{l,q,s}$. This latter varies with the constellation order and the position of the symbol carried by the path. To illustrate this, refer again to Fig. 3 with $R_s = 1 \cdot d_0$. If the winning path carries a symbol at the center of the constellation, a total of 5 L-values will have to be updated. However, if the carried symbol is located at the corner of the constellation, only 3 updates will have to be performed. Note that the L-value of the symbol carried by the winning path also needs to be updated. Having a varying level of complexity reduction depending on the location within the merge tree merge operations on one side and on the location of the symbol carried by the winning path on the other, we provide bounds on the the complexity analysis. They depend on the value of R_s^l (a design choice) and the 2 extreme possible position values of the winning path within the constellation as shown for the case of $q = 64$ in Tab. I.

Table I: Boundary values of $\lambda_{q,s}$ for different R_s with $q = 64$.

R_s	1	2	3	4	5	6
Max $\lambda_{q,s}$	5	13	29	47	63	64
Min $\lambda_{q,s}$	3	6	11	17	26	35

We can then bound the complexity in terms of number of updates for the decoding of one symbol as:

$$C_{tot} = \sum_{l=1}^{\log_2(q)} N_{l,q}\lambda_{l,q,s} = \sum_{l=1}^{\log_2(q)} q\frac{\lambda_{l,q,s}}{2^l} \tag{10}$$

Reducing the radius size R_s^l has a noticeable impact on the error correcting performance of the decoder. Therefore, we evaluate a number of benchmark configurations in the next section before comparing the complexity of the proposed algorithm with state of the art in Section VII.

VI. Simulation Results

We consider NB-TCs over GF(64) mapped to a 64-QAM constellation, designed from rate-1/2 NB-CCs as in Fig. 1-(a). The coefficients a_1, a_2 and a_3 were chosen to achieve the highest minimum cumulated Euclidean distance, and are equal to 31, 5 and 18, respectively. To construct the finite field, we used the following primitive polynomial: $P_{\text{GF}(64)}(D) = 1 + D^2 + D^3 + D^5 + D^6$. The Turbo code interleaver uses an Almost Regular Permutation (ARP) designed for the frame size of $K_s = 160$ GF(64) symbols (960 bits) following [17]–[19]. To assess the proposed NN-LSOVA algorithm, several low-complexity configurations were evaluated (see Table II). These configurations use different radii for different LSOVA layers l, leading to a layer-dependent complexity.

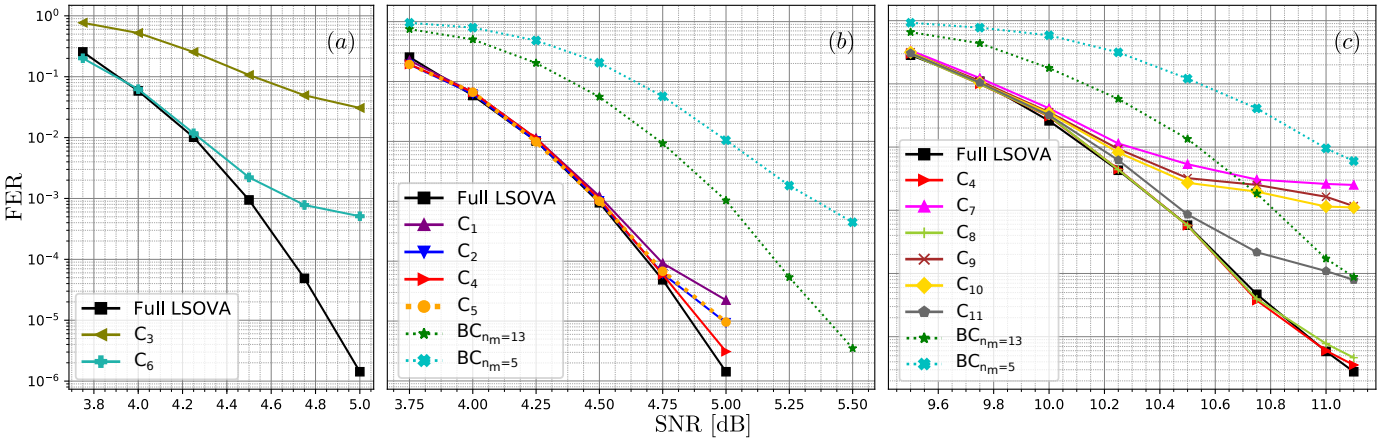

Figure 4: FER curves after 8 decoding iterations of: (a) Full LSOVA, C_3 and C_6 (b) Full LSOVA, C_1, C_2, C_4, C_5 and BC with $n_m = 13$ and $n_m = 5$ (c) Full LSOVA, C_4, C_7-C_{11}, and BC with $n_m = 13$ and $n_m = 5$. Rate: (a) 1/3, (b) 1/3, (c) 4/5.

1) Simulation results for rate 1/3: Fig. 4 (a) compares the FER performance at the output of a decoder with full LSOVA ($\equiv$ full MLM) on one side and NN-LSOVA using a uniform radius of 3 and 5 across all merge layers (configurations C_3, C_6 in Table II) on the other. Clearly, the uniform reduction of R_s^l leads to an undesired error floor at a FER of about $3 \cdot 10^{-4}$ for C_6 while the decoder fails completely for C_3. Recall that the merge operations are performed in a binary tree, which suggests that the impact of a reduced radius varies in between layers. Therefore, Fig. 4 (b) compares NN-LSOVA configurations with non-uniform reductions in R_s^l. Overall an improvement of the FER performance can be observed when moving from a uniform reduction of R_s^l to a non-uniform reduction. Performing the last two merge layers with a radius of $R_s^6 = 6$ leads to a better performance, even when reducing the radius more in the first layers (C_1 vs. C_2). Comparing C_2, C_4 and C_5 reveals that the NN-LSOVA is more sensitive to a reduction of R_s in the first layer than in the middle layers. C_4 shows the best performance.

2) Simulation results for rate 4/5: As mentioned in section IV, the probability of error decreases exponentially with the coding gain. For a similar target error rate to rate 1/3, the rate 4/5 suffers from a lower coding gain mainly due to the considerable decrease in the minimum cumulated Euclidean distance due to puncturing. Hence a larger operating SNR is required or equivalently, only a lower AWGN variance can be supported. This is expected to reduce the entailed penalty from considering smaller R_s^l values. Simulation results shown in Fig. 4 (c) confirm this expectation. Not only does C_4 yield Full-LSOVA performance, but a further reduction as for C_8 is possible without FER loss. However, results for C_7, C_9, C_{10} and C_{11} demonstrate that a radius reduction down to 1 even for the middle layers is to be avoided.

VII. Complexity Comparison

With the performance obtained in the previous section in mind, we provide here a complexity analysis for the simulated configurations based on Table I and Eq. (10). The complexity reduction of the NN-LSOVA decoding in comparison with the *full* LSOVA decoding (i.e. with full radius R_s^l for each layer) is illustrated in Table II, for the best and worst case scenarios, i.e. when $\lambda_{l,q,s}$ is at its minimum or maximum value.

The viability of using NN-LSOVA is confirmed by a complexity reduction of up to 52% in terms of number of updates for the worst case scenario for the C_4 configuration with FER performance competitive to Full-LSOVA/MLM for rate 1/3 and 4/5 and up to almost 90% for configuration C_8 for rate 4/5. However, in order to allow a fair comparison with the MLM algorithm, we need to express the complexity in terms of ACS (Add Compare Select) operations. Note, that the NN-LSOVA only affects the complexity for the extrinsic computation. Each L-value update in the SOU of the LSOVA/NN-LSOVA is done using Eq. (8), requiring at most one ACS operation. Every merge therefore needs $\lambda_{l,q,s}$ ACS operations plus one comparison between the path metrics to find the winning path, which will be counted as half an ACS operation. For the decoding of one frame, we get:

$$\text{ACS}_{\text{LSOVA}} = n_{enc} \cdot K_s \cdot n_{it} \cdot \sum_{l=1}^{\log_2(q)} \frac{q}{2^l} \left(\lambda_{l,q,s} + \frac{1}{2} \right) \quad (11)$$

where n_{enc} is the number of component encoders in the NB-TC structure, K_s is the number of GF(q) symbols in the frame

Table II: Selected NN-LSOVA configurations, number of updates and complexity comparison for best-/worst case.

Configuration ($R_s^1 R_s^2 R_s^3 R_s^4 R_s^5 R_s^6$)	Updates (bc)	Compl. red. (bc)	Updates (wc)	Compl. red. (wc)
C_1 = (3,4,4,5,5,6)	951	76.41%	2498	38.05%
C_2 = (2,3,4,5,6,6)	713	82.32%	1700	57.84%
C_3 = (2,2,2,2,2,2)	378	90.63%	819	79.69%
C_4 = (3,3,3,3,6,6)	765	**81.03%**	1932	**52.08%**
C_5 = (2,3,3,6,6,6)	701	82.61%	1560	61.31%
C_6 = (5,5,5,5,5,5)	1638	59.38%	3969	1.56%
C_7 = (3,1,1,6,6,6)	669	83.41%	1496	62.90%
C_8 = (2,2,2,2,4,6)	429	**89.36%**	938	**76.74%**
C_9 = (2,1,2,2,4,6)	381	90.55%	810	79.91%
C_{10} = (2,2,1,2,4,6)	405	89.96%	874	78.32%
C_{11} = (2,2,2,1,4,6)	417	89.66%	906	77.53%

and n_{it} is the number of performed decoding iterations. As a point of comparison, we take the BC-based method from [8] for its different approach for complexity reduction in decoding NB-TCs. Its complexity for the extrinsic computation is:

$$\text{ACS}_{\text{BC}} = n_{enc} \cdot K_s \cdot n_{it} \cdot 2n_m \cdot q \tag{12}$$

Where n_m is a parameter of the algorithm that denotes the size of the used sorted tables. The number of ACS operations in the extrinsic computation of the MLM is computed by

$$\text{ACS}_{\text{MLM}} = n_{enc} \cdot K_s \cdot n_{it} \cdot q^2 \tag{13}$$

Based on Eqs. (11), (12) and (13), Table III compares the complexity in terms of performed ACS operations for the extrinsic computation. A value of $n_m = 25$ is assumed for the BC, which results in a FER performance close to MLM. Furthermore, $n_{enc} = 2$, $K_s = 160$ and $n_{it} = 8$.

Table III: No. of ACS operations for MLM, BC and NN-LSOVA and complexity reduction compared to MLM.

Alg.	Num. of ACS Op.		Compl. red. [%]	
MLM	10,485,760		0	
BC [8]	8,192,000		21.24	
NN-LSOVA Config.	worst	best	worst	best
C_1	6,475,520	2,515,200	**38.24**	76.01
C_2	4,432,640	1,905,920	57.73	81.82
C_3	2,177,280	1,048,320	79.23	90.00
C_4	5,026,560	2,039,040	52.06	80.55
C_5	4,074,240	1,875,200	61.15	82.12
C_6	10,241,280	4,273,920	2.33	59.2
C_7	3,910,400	1,793,280	62.71	82.90
C_8	2,466,560	1,163,520	76.48	88.90
C_9	2,154,240	1,056,000	**79.46**	89.93
C_{10}	2,318,080	1,117,440	77.89	89.34
C_{11}	2,400,000	1,148,160	77.53	89.66

With the proposed NN-LSOVA, we achieve between around 40% and 80% of complexity reduction in comparison to the MLM (worst case, not considering C_6 which has insufficient FER performance), which corresponds to an additional complexity reduction of between around 17% and 58% compared to the BC. In particular, a complexity reduction of at least 52.06% is achieved for configuration C_4, which is competitive in terms of FER performance compared to the MLM and the BC for the two considered rates of 1/3 and 4/5. To obtain the same low complexity with the BC, we would need to have $n_m = 13$ for the worst case, and $n_m = 5$ for the best case, significantly penalizing the FER performance (see Fig. 4)

VIII. Conclusion

In this work, we applied for the first time the LSOVA algorithm to the decoding of NB-Turbo Codes. Moreover for a direct mapping between the GF(q) code and constellation symbols, we proposed a new reduced complexity decoding algorithm. Denoted by NN-LSOVA, it achieves a complexity reduction by up to **52%** in terms of performed ACS operations compared to the MLM while maintaining similar FER performance. It can even reach up to almost **80%** for applications targeting only high code rates or FER larger than 10^{-4}. Furthermore for the same target performance, it was shown to outperform the prior art solution based on the BC by up to more than 30%. These results show the NN-LSOVA algorithm to be a promising path towards efficient hardware implementations for NB Turbo decoding, and future works will consider combining the BC for state metric computation with the NN-LSOVA for the soft output.

Acknowledgment

This work was partially funded by the French National Research Agency projects QCSP (ANR-19-CE25-0013-01) and TurboLEAP (ANR-20-CE25-0007).

References

[1] C. Berrou, A. Glavieux, and P. Thitimajshima. Near Shannon limit error-correcting coding and decoding: Turbo-codes. In *IEEE Intern. Conf. on Commun. (ICC)*, volume 2, pages 1064–1070, Geneva, Switzerland, May 1993.

[2] D. J. C. MacKay and R. M. Neal. Near Shannon limit performance of low density parity check codes. *Electron. Lett.*, 33(6):457–458, 1997.

[3] G. Liva, E. Paolini, B. Matuz, S. Scalise, and M. Chiani. Short turbo codes over high order fields. *IEEE Trans. on Commun.*, 61(6):2201–2211, 2013.

[4] R. Klaimi, C. Abdel Nour, C. Douillard, and J. Farah. Design of Low-Complexity Convolutional Codes over GF(q). In *IEEE Global Commun. Conf. (GLOBECOM)*, Abu Dhabi, United Arab Emirates, Dec 2018.

[5] M. C. Davey and D. MacKay. Low-density parity check codes over GF(q). *IEEE Commun. Lett.*, 2(6):165–167, 1998.

[6] L. Barnault and D. Declercq. Fast decoding algorithm for LDPC over GF(2^q). In *IEEE Inf. Theory Workshop*, pages 70–73, 2003.

[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.*, 46(9):633–634, 2010.

[8] R. Klaimi, C. A. Nour, C. Douillard, and J. Farah. Low-complexity decoders for non-binary turbo codes. In *10th Intern. Symp. on Turbo Codes Iter. Inf. Proc. (ISTC)*, pages 1–5, Hong Kong, China, Dec. 2018.

[9] V. H. S. Le, C. Abdel Nour, E. Boutillon, and C. Douillard. Revisiting the Max-Log-Map Algorithm With SOVA Update Rules: New Simplifications for High-Radix SISO Decoders. *IEEE Trans. on Commun.*, 68(4):1991–2004, 2020.

[10] S. Weithoffer, R. Klaimi, C. Abdel Nour, N. Wehn, and C. Douillard. Low-complexity Computational Units for the Local-SOVA Decoding Algorithm. In *IEEE 31st Intern. Symp. on Personal, Indoor and Mobile Radio Commun. (PIMRC)*, London, UK, Sept. 2020.

[11] P. Robertson, E. Villebrun, and P. Hoeher. A Comparison of Optimal and Sub-Optimal MAP decoding Algorithms Operating in the Log-Domain. In *IEEE Intern. Conf. on Commun. (ICC)*, pages 1009–1013, Seattle, Washington, USA, June 1995.

[12] J. Berkmann. On turbo decoding of nonbinary codes. *IEEE Commun. Lett.*, 2(4):94–96, 1998.

[13] J. Hagenauer and P. Hoeher. A Viterbi algorithm with soft-decision outputs and its applications. In *IEEE Global Telecommun. Conf. and Exhibition*, pages 1680–1686 vol.3, Dallas, TX, USA, Nov 1989.

[14] G. Battail. Pondération des symboles décodés par l'algorithme de Viterbi. In *Annales des telecom.*, volume 42, pages 31–38. Springer, 1987.

[15] Lang Lin and R. S. Cheng. Improvements in SOVA-based decoding for turbo codes. In *IEEE Intern. Conf. on Commun. (ICC)*, volume 3, pages 1473–1478 vol.3, June 1997.

[16] R. Klaimi, C. Abdel Nour, C. Douillard, and J. Farah. Union bound evaluation for non-binary turbo coded modulations. *IEEE Communications Letters*, 24(6):1178 – 1182, 2020.

[17] R. Garzón-Bohórquez, C. Abdel Nour, and C. Douillard. Improving turbo codes for 5G with parity puncture-constrained interleavers. In *9th Intern. Symp. on Turbo Codes Iter. Inf. (ISTC)*, pages 151–155, Brest, France, Sep. 2016.

[18] R. Garzón Bohórquez, C. Abdel Nour, and C. Douillard. Protograph-based interleavers for punctured turbo codes. *IEEE Trans. Commun.*, 66(5):1833–1844, May 2018.

[19] R. Garzón Bohórquez, R. Klaimi, C. A. Abdel Nour, and C. Douillard. Mitigating correlation problems in turbo decoders. In *10th Intern. Symp. on Turbo Codes Iter. Inf. (ISTC)*, Hong Kong, China, December 2018.