Simplified Compression of Redundancy Free Trellis Sections in Turbo Decoder

Emmanuel Boutillon, José-Luis Sanchez-Rojas, Cédric Marchand

Simplified Compression of Redundancy Free Trellis Sections in Turbo Decoder

Emmanuel Boutillon[⊥], *Senior Member, IEEE*, José-Luis Sanchez-Rojas[*], Cédric Marchand[⊥], *Member, IEEE*
[⊥]Lab-STICC, UMR 6582, Université de Bretagne Sud, 56100 Lorient, France
[*]INICTEL-UNI, av. San Luis 1771, San Borja, Lima 41, Lima, Peru

***Abstract*—It has been recently shown that a sequence of $R = q(M-1)$ redundancy free trellis stages of a recursive convolutional decoder can be compressed in a sequence of $L = M-1$ trellis stages, where M is the number of states of the trellis and q is a positive integer. In this paper, we show that for an M state Turbo decoder, among the L compressed trellis stages, only $m = 3$ or even $m = 2$ are necessary. The so-called m-min algorithm can either be used to increase the throughput for decoding a high rate turbo-code and/or to reduce its power consumption.**[1]

I. Introduction

The quality of an error control code design can be evaluated in terms of decoding performance, implementation cost (area, power dissipation) and decoding throughput. A general overview of error control code decoders can be found in [1]. The recent specifications of wireless systems (LTE, HSPA [2]) propose the use of turbo-codes with very high code rates (typically, between 0.8 and 0.98). In contrast to code construction, there are very few papers dedicated to decoders with such high rates. Most of the reported architectures propose the basic structure of a rate 1/3 decoder, with parameters (i.e. window lengths) optimized for high rates to tradeoff performance and decoding throughput. In [3], a method is proposed to directly exploit the existence of long sequences (up to 100) of Redundancy Free Trellis Sections (RFTS, or sequences of bits without redundancy) to reduce the complexity of part of the decoder: during the acquisition process, any RFTS of size R is replaced by a shorter RFTS sequence of size $L = M-1$, with additional $(R \mod L)$ steps of shuffling. In this paper, we show that, in the context of a Max-Log-MAP decoder [5], among the L steps of the compressed RFTS, only $m = 3$ or $m = 2$ are really useful, which allows further architectural optimization.

The remainder of the paper is divided into four sections. Sec. II gives enough information about the trellis compression to have a self-consistent paper. Then, Sec. III presents the sub-optimal 3-min simplification; followed in Sec. IV by a discussion about hardware implementation. Finally, Sec. V concludes the paper.

II. Principle of the RTFS trellis compaction

In this section, we will first recall the problem of acquisition for high rate turbo-codes. Then, we will present the principle of trellis compaction at the encoding level before deriving it at the decoding level.

A. Acquisition for high rate turbo-codes

The implementation of a turbo-code is a well investigated area. The standard implementation uses the Log-MAP algorithm or the Max-Log-MAP algorithm [5] along with the sliding window algorithm [6]. This algorithm consists in dividing the frame of length K into windows of length W and processing the forward-backward steps on a W-sized block instead of a whole K-sized block. To process the p^{th} window, an accurate estimation of the initial forward state metrics α_{pW} and backward state metrics β_{pW+W} are required. One possible scheduling is to perform the backward recursion directly from index K down to 1 to obtain naturally the initial β_{pW+W} states and to obtain the α_{pW} initial states by an acquisition of size W', starting from state $\alpha^0_{pW-W'}$ to state α_{pW} (see Fig. 1). This pre-processing is called "acquisition". The initial value $\alpha^0_{pW-W'}$ can be either the all-zero state vector (if there is no a priori knowledge on the initial state) or a forward state vector stored from the previous iteration. This last method is commonly called Next Iteration Initialization (NII) [7].

For high code rates, the length of the acquisition L' needs to be high enough to contain some non RFTS (i.e. trellis sections associated with non-punctured redundancy bits). In fact, starting from the all-zero state vector, if the acquisition processes only RFTS, the final state will also be the all-zero state vector and the acquisition process will thus be

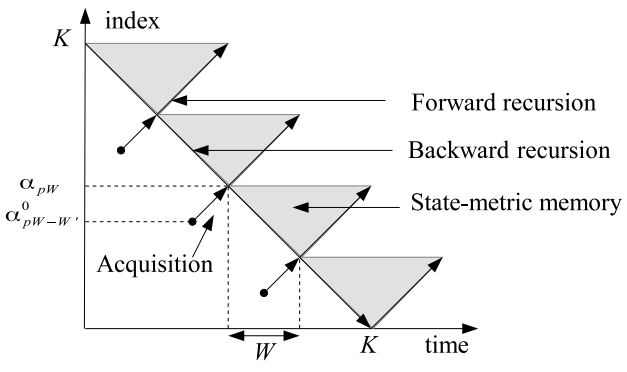

Fig. 1. Schematic representation of the sliding-window algorithms, with parameters $W = K/4$ and $W' = W/2$. The x-axis represents the time index and the y-axis the index of the bit.

[1]This work has been supported by the GIGADEC project from Brittany region, as well as the CPER project PALMYRE II (Brittany region and FEDER funding).

useless. By simulation, it is verified that a high value of L' is required, even if the NII technique is used. This means that the acquisition step consumes a significant portion of the decoder's processing effort, which reduces the efficiency and throughput of the hardware implementation.

B. Compression of RFTS at the encoding level

Let us consider the trellis compression in the case of hard information bits (a decision is made on the value of the received bit, i.e., no soft value is available). For an 8-state convolutional encoder, the state-space representation of [8] gives

$$\mathbf{X}_{k+1} = \mathbb{A}\mathbf{X}_k + \mathbb{B}D_k, \tag{1a}$$

$$\mathbf{V}_k = \mathbb{C}\mathbf{X}_k + \mathbb{D}D_k, \tag{1b}$$

where $\mathbf{X}_k$, D_k and V_k are respectively the state of the encoder, the input information bit and the output vector at time k. The matrices $(\mathbb{A}, \mathbb{B}, \mathbb{C}, \mathbb{D})$ are respectively the state matrix, the entry matrix, the output matrix and the feedforward matrix. In (1a) and (1b), arithmetic is over GF(2). Moreover, for a recursive code, the matrix $\mathbb{A}$ verifies $\mathbb{A}^L = \mathrm{Id}$, where Id is the identity matrix. Fig. 2.a shows the structure of the encoder (the generation of output bits, corresponding to (1.b), is omitted). Starting from a state $\mathbf{X}_0$ and the bit sequence $D_k, k = 0, 1, \ldots, R-1$, the final state of the encoder is given by

$$\mathbf{X}_R = \mathbb{A}^R \cdot \mathbf{X}_0 + \sum_{k=0}^{R-1} \mathbb{A}^k \cdot \mathbb{B} \cdot D_{R-1-k}. \tag{2}$$

Let us replace index k with $qL + l$, where $l = k \mod L$ and $q = \frac{k-l}{L}$. For the sake of simplicity, $k \mod L$ will be denoted $k_{|L}$. Then (2) can be rewritten as

$$\mathbf{X}_R = \mathbb{A}^R\mathbf{X}_0 + \sum_{l=0}^{L-1} \sum_{q=0}^{\lfloor \frac{R-1-l}{L} \rfloor} \mathbb{A}^{qL+l}\mathbb{B}D_{R-1-qL-l}. \tag{3}$$

Since $\mathbb{A}^L = \mathrm{Id}$, $\mathbb{A}^{qL+l} = \mathbb{A}^l$, thus (3) can be expressed as:

$$\mathbf{X}_R = \mathbb{A}^{R_{|L}}\mathbf{X}_0 + \sum_{l=0}^{L-1} \mathbb{A}^l\mathbb{B} \left(\sum_{q=0}^{\lfloor \frac{R-1-l}{L} \rfloor} D_{R-1-qL-l} \right), \tag{4}$$

where $R_{|L} = R \mod L$. We perform the change of variable $p = \lfloor \frac{R-1-l}{L} \rfloor - q$ in the last summation term of (4). Let $h(l) = (R-1-l) \mod L$. We have

$$\sum_{q=0}^{\lfloor \frac{R-1-l}{L} \rfloor} D_{R-1-qL-l} = \sum_{p=0}^{\lfloor \frac{R-1-h(l)}{L} \rfloor} D_{h(l)+pL}. \tag{5}$$

Let us give an example to illustrate (5) with $R = 23$, $L = 7$ and $l = 3$. The left term of (5) gives $D_{19} + D_{12} + D_5$. Since $h(3) = (23 - 1 - 3) \mod 7 = 5$, the right term of (5) gives $D_5 + D_{12} + D_{19}$ and both terms are equal. Let D_h^a be the "aggregated bit" corresponding to the bits having the same residue $h \mod L$: for $h = 0, 1, \ldots, L-1$

$$D_h^a = \sum_{p=0}^{\lfloor \frac{R-1-h}{L} \rfloor} D_{h+pL}. \tag{6}$$

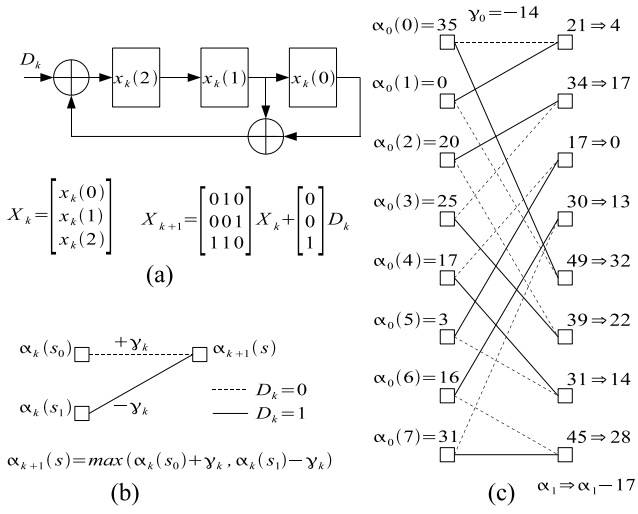

Fig. 2. (a) 8-state LTE encoder structure, (b) Step of the trellis (c) Example of RFTS to compute α_1 from α_0 and γ_0 (see table I).

Note: when $h \geq L$, D_h^a is equal to $D_{h \mod L}^a$. Then, the last summation of (4) can be expressed as $D_{h(l)}^a$. Let $\mathbf{X}'_0 = \mathbb{A}^{R_{|L}}\mathbf{X}_0$. Using the fact that $\mathbb{A}^L\mathbf{X}'_0 = \mathbb{A}^{R_{|L}}\mathbf{X}_0$, (4) can be expressed recursively starting from state $\mathbf{X}'_0$ to state $\mathbf{X}'_L = \mathbf{X}_R$ as, for $l = 0, 1, \ldots, L-1$

$$\mathbf{X}'_{l+1} = \mathbb{A}\mathbf{X}'_l + \mathbb{B}D^a_{h(L-1-l)}. \tag{7}$$

Since $h(L-1-l) = (R-1-(L-1-l)) \mod L = (R_{|L}+l) \mod L$, (7) can be expressed simply as:

$$\mathbf{X}'_{l+1} = \mathbb{A}\mathbf{X}'_l + \mathbb{B}D^a_{R_{|L}+l}. \tag{8}$$

Note that (8) has the same structure as (1a). Moreover, the computation of $\mathbf{X}'_0 = \mathbb{A}^{R_{|L}}\mathbf{X}_0$ can be performed in $R_{|L}$ steps of (1a) with dummy null input bits $D_k^d = 0, k = 0, 1, \ldots, R_{|L} - 1$. In other words, every RFTS sequence of length R (i.e. R trellis stages with no redundancy) can be reduced to an RFTS sequence of length $R_{|L} + L$. The first $R_{|L}$ RFTS are associated with a null dummy bit $D_k^d = 0$, $k = 0, 1, \ldots, R_{|L}-1$ and the last L trellis stages are associated with the aggregated bit $D_l^a, l = R_{|L}, \ldots, L-1+R_{|L}$. One should note that shifting the position of an aggregated bit by a multiple of L does not affect the result. Swapping the sub-block of the last $R_{|L}$ aggregated bits with the sub-block of the first $R_{|R}$ dummy bits leads to another valid compaction of the trellis. In that case, the last $R_{|L}$ stages are associated with dummy null bits. This solution was originally presented in [3].

C. Trellis compaction

Let us recall the classical forward recursion of the MAP decoder. Let α_k be the forward state metrics at time k and γ_k the branch metrics at time k. The recursion is given by

$$\alpha_{k+1}(s) = \max^*(\alpha_k(s_0) + \gamma_k(s_0, s), \alpha_k(s_1) + \gamma_k(s_1, s)), \tag{9}$$

where s is the current state, (s_0, s) is the branch in the trellis associated with the input bit $D_k = 0$ connecting state s_0 to the

state s. Similarly, (s_1, s) is the branch in the trellis associated with the input bit $D_k = 1$ connecting state s_1 to state s. The $\max^*$ function is defined as

$$\max^*(a, b) = \max(a, b) + e^{-|a-b|}. \quad (10)$$

When the Max-Log-MAP algorithm is used, the $\max^*$ operator in (9) is simply replaced by the $\max$ operator. Fig. 2.b shows the computation of the state metric $\alpha_{k+1}(s)$.

For an RFTS, there is no associated redundancy bit and the branch metrics are simply $\gamma(s_0, s) = \mathrm{LLR}(D_k)$ and $\gamma(s_1, s) = -\mathrm{LLR}(D_k)$, where $\mathrm{LLR}(D_k)$ is the Log-Likelihood Ratio of the received bit, i.e. $\mathrm{LLR}(D_k) = \ln\left(\frac{P(D_k=0)}{P(D_k=1)}\right)$. In the sequel, for the sake of simplicity, $\mathrm{LLR}(D_k)$ will be denoted γ_k. Similarly, γ_l^a will represent the LLR of the aggregated bit D_l^a and γ_l^d the LLR of the dummy null bit ($\gamma_l^d = +\infty$ since $D_l^d = 0$).

As shown in Sec. II.B, at the encoder side, a sequence of length R can be reduced to a sequence of length $R_{|L}$ with dummy zero inputs and a sequence of length L of aggregated bits. At the receiver side, this implies that the RFTS sequence can be compressed into $R_{|L}$ steps of shuffling (using the branch metric $\gamma_l^d = +\infty$), followed by L trellis sections of aggregated bits γ_l^a. Since summation in (6) is over GF(2), at the receiver side, $\gamma_l^a = \mathrm{LLR}(D_l^a)$ is given by

$$\gamma_l^a = 2\tanh^{-1}\left(\prod_{q=0}^{\lfloor\frac{R-1-l}{L}\rfloor}\tanh\left(\frac{\gamma_{qL+l}}{2}\right)\right). \quad (11)$$

Thanks to the min-sum approximation, (11) can be simplified by separating the modulus of γ_l^a and the sign of γ_l^a:

$$|\gamma_l^a| = \min\left\{|\gamma_{qL+l}|, q = 0, 1, \ldots, \left\lfloor\frac{R-1-l}{L}\right\rfloor\right\}, \quad (12)$$

$$\mathrm{sign}(\gamma_l^a) = \prod_{q=0}^{\lfloor\frac{R-1-l}{L}\rfloor}\mathrm{sign}(\gamma_{qL+l}). \quad (13)$$

D. Example of RFTS compaction

Let us consider the $M = 8$-state encoder defined in Fig. 2.a, a RFTS sequence of length $R = 16$ with initial state metrics at time 0 equal to $\alpha_0 = [35, 0, 20, 25, 17, 3, 16, 31]'$. The LLRs of the D_k values are equal to $\{\gamma_k\}_{k=0,1,\ldots,15} = \{-14,\ 31,\ 24,\ 12,\ 31,\ 20,\ 6,\ -31,\ 19,\ 15,\ -19,\ -8,\ 15,\ 12,\ 5,\ -11\}$[2]. Let us assume an RFTS sequence of length $R = 16$. Table I shows the first forward state metrics and the last forward state metrics of the RFTS sequence. In order to bound the increasing values of the state metrics, at each stage, the minimum value of the state metrics are subtracted out ($\alpha_k = \alpha_k - \min(\alpha_k(i), i = 0, 1, \ldots, 7)$). After this subtraction, the minimum metric is always equal to zero. Fig. 2.c shows the details of the computation of α_1 from α_0 and γ_0. Let us compute γ_2^a. From (12), we obtain $|\gamma_2^a| = \min\{|D_2|, |D_9|\} = \min\{|24|, |15|\} = 15$. From (13), we obtain $\mathrm{sign}(\gamma_2^a)) = 1$. The computation for the other values gives $\{\gamma_l^a\}_{l=2,3,\ldots,8} = \{15, -12, -8, 15, 6, 5, -11\}$. Table II shows the 9 steps of the compressed trellis. As expected, the last stage of the classical trellis (last column of Table I) and the last stage of the compressed trellis are equal (last column of Table II). This algorithm implies the use of L minima. It is called the L-min algorithm. In terms of performance, processing acquisition sequences of the sliding window algorithm without or with trellis compression gives identical final results. In the latter case, the number of clock cycles can be significantly reduced, leading to a more efficient architecture, as described briefly in [3]. Now the question arises whether it is possible to further decrease the complexity of the L-min algorithm by trading-off complexity and performance.

TABLE I
CLASSICAL TRELLIS FOR $R = 16$

k	0	1	2	-	11	12	13	14	15	16
γ_k	-14	31	24	-	-8	15	12	5	-11	...
$\alpha_k(0)$	35	4	4	-	13	18	18	18	6	16
$\alpha_k(1)$	0	17	13	-	22	0	24	13	16	12
$\alpha_k(2)$	20	0	32	-	4	12	9	0	12	6
$\alpha_k(3)$	25	13	28	-	20	24	13	28	4	2
$\alpha_k(4)$	17	32	17	-	32	9	0	24	1	6
$\alpha_k(5)$	3	22	0	-	0	16	12	9	6	4
$\alpha_k(6)$	16	14	22	-	28	28	16	12	2	1
$\alpha_k(7)$	31	28	14	-	17	13	28	16	0	0

TABLE II
COMPRESSED TRELLIS (L-MIN ALGORITHM)

	γ_l^d		γ_l^a							
l	0	1	2	3	4	5	6	7	8	9
	$+\infty$	$+\infty$	15	-12	-8	15	6	5	-11	...
$\alpha_l(0)$	35	35	35	35	13	18	18	18	6	16
$\alpha_l(1)$	0	25	31	16	22	0	24	13	16	12
$\alpha_l(2)$	20	17	0	25	4	12	9	0	12	6
$\alpha_l(3)$	25	31	16	3	14	24	13	28	4	2
$\alpha_l(4)$	17	0	25	31	32	9	0	24	1	6
$\alpha_l(5)$	3	20	17	0	0	10	12	9	6	4
$\alpha_l(6)$	16	3	20	17	28	28	10	12	2	1
$\alpha_l(7)$	31	16	3	20	17	13	28	16	0	0

III. m-MIN$_a$ AND m-MIN$_g$ COMPUTATION

The analysis of the Max-Log-MAP algorithm for a sequence of RFTS shows that a high magnitude of γ implies simply a shuffling of the state metrics value α, while a low magnitude of γ decreases significantly the dynamics of the α terms (i.e. a low reliability bit gives more uncertainty on the current state of the encoder). This indicates that it may be sufficient to consider only a small subset of the initial γ values to process the RFTS sequence. Let us define the m-min$_a$ method as an extension of the L-min method where among the L LLRs of the aggregated bits, the $L-m$ highest γ^a modules are saturated to $\mathrm{sign}(\gamma)\times\infty$. Using this method with the example of the previous section, the 3-min$_a$ method now consists of replacing the aggregated γ^a values $\{15, -12, -8, 15, 6, 5, -11\}$ by keeping only the 3 smallest magnitude values and saturating the others, i.e., the set $\{+\infty, -\infty, -8, +\infty, 6, 5, -\infty\}$. Table III shows the corresponding trellis. Compared to Table II, intermediate state metrics can differ from the L-min and the 3-min methods, but finally, both methods lead to the same final state. In the

[2]The MATLAB code used for this example can be downloaded at [9]

TABLE III
COMPRESSED TRELLIS (3-MIN ALGORITHM)

	γ_l^d		γ_l^a							
l	0	1	2	3	4	5	6	7	8	9
$\gamma_l^{d,a}$	∞	$+\infty$	∞	-∞	-8	∞	6	5	$-\infty$	...
$\alpha_l(0)$	35	35	35	35	16	24	24	18	6	16
$\alpha_l(1)$	0	25	31	16	25	0	30	13	16	12
$\alpha_l(2)$	20	17	0	25	0	18	15	0	12	6
$\alpha_l(3)$	25	31	16	3	17	30	19	28	4	2
$\alpha_l(4)$	17	0	25	31	35	15	0	24	1	6
$\alpha_l(5)$	3	20	17	0	3	16	18	9	6	4
$\alpha_l(6)$	16	3	20	17	31	34	16	12	2	1
$\alpha_l(7)$	31	16	3	20	20	19	34	16	0	0

TABLE IV
REQUIRED E_b/N_o TO OBTAIN A FER OF 10^{-2}. HSPA TURBO-CODE OF LENGTH $K = 5144$ FOR SEVERAL CODE RATES. FB = FORWARD-BACKWARD, SW = SLIDING WINDOW, ACQUISITION WITH NII AND $W = W'$

rate	$W' = W$	FB	FB-SW	2-min$_g$	1-min$_g$
$r = 0.8$	32	3.89 dB	3.92 dB	3.94 dB	4.05 dB
$r = 0.9$	64	4.87 dB	4.89 dB	4.91 dB	5.02 dB
$r = 0.94$	128	5.49 dB	5.51 dB	5.53 dB	5.71 dB
$r = 0.98$	32	6.73 dB	8.17 dB	8.18 dB	8.20 dB
	64	6.73 dB	7.09 dB	7.10 dB	7.15 dB
	128	6.73 dB	6.83 dB	6.84 dB	6.92 dB
	256	6.73 dB	6.75 dB	6.77 dB	6.87 dB

general case, the final state metrics can differ slightly, even using the 4-min or 5-min methods.

In practice, the only pertinent criterion to evaluate an algorithm is the performance loss. To this end, we have run several simulations performing all acquisitions using the m-min$_a$ algorithm. We also tested a modified version of the m-min$_a$ method, performing the saturation before the aggregation of the bits. In this case, the $R-m$ highest values of the RFTS sequence are saturated prior to the aggregation method. This variation of the m-min$_a$ method is called m-min$_g$ method. Note that the m-min$_a$ and the m-min$_g$ methods lead to the same result if $m = 1$. For $m > 1$, the two methods give the same result only if the m minimum values before aggregation have all distinct indices modulo L.

Table IV shows bit-true simulation results of an HSPA Turbo decoder with $K = 5144$ using the sliding window technique associated with the NII technique. For $r = 0.98$, various acquisition lengths W' are given to illustrate the need of high values of W' for very high code rate. In all cases, the 3-min algorithm (not shown in the table) does not have significant performance degradation. The 2-min$_g$ and 1-min$_g$ algorithms degrade the performance by less than 0.02 dB and 0.2 dB respectively for all code rates.

IV. ARCHITECTURE OPTIMIZATION

In [3], the authors proposed to perform the bit aggregation "on the fly" during the previous iteration. The time for processing the acquisition is thus reduced and the decoding throughput is increased. In this paper, we present new applications of RFTS compression to reduce the power consumption of the decoder. Let us focus on the 2-min$_g$ aggregation in an RFTS acquisition sequence of length R. The RFTS compression can be done on the fly in three steps. Step one: during the first $R - (R_{|L}) - L$ clock cycles, the acquisition forward unit is frozen (and thus saving energy). Step two: during the next $R_{|L}$ clock cycles, permutations of state metrics are done (processing of $R_{|L}$ dummy bits). Step 3: during the last L clock cycles, the forward unit is processed with the aggregated bit computed on the fly by algorithm 1. For $R = 100$ (code rate 0.98), this method allows the decoder to freeze the acquisition forward unit 90% of the time while the low complexity bit aggregation algorithm works. This method does not increase the throughput but helps to save power dissipation.

Initialisation: $\text{sign}(l) = +1$, for $l = 0, 1, \ldots, L-1$;
$(\texttt{min}_1, \texttt{ind}_1) = (+\infty, 0)$; $(\texttt{min}_2, \texttt{ind}_2) = (+\infty, 1)$
for $k = 0$ *to* $R-1$ **do**
  $l = k \mod L$;
  $\text{sign}(l) = \text{sign}(l) \times \text{sign}(\gamma_k)$
  if $|\gamma_k| < \texttt{min}_1$ **then**
    $\texttt{min}_2 = \texttt{min}_1$, $\texttt{min}_1 = |\gamma_k|$
    $\texttt{ind}_2 = \texttt{ind}_1$, $\texttt{ind}_1 = l$
    else if $|\gamma_k| < \texttt{min}_2$ **then**
      $\texttt{min}_2 = |\gamma_k|$, $\texttt{ind}_2 = l$
end

Algorithm 1: On the fly aggregation of the bit for the 2-min$_g$ algorithm.

V. CONCLUSION

In this paper, we showed that the compression of redundancy free trellis stages using the L-min algorithm can be further simplified by considering only 3, or even 2, minima among the L values. Simulations on 3-min show no performance loss and simulation on 2-min shows minor performance loss (around 0.02 dB). The 3-min (or 2-min) algorithm opens new architecture optimization, either to save power during the acquisition or to increase the overall decoding throughput.

REFERENCES

[1] F. Kienle, N. Wehn, H. Meyr, "On Complexity, Energy- and Implementation-Efficiency of Channel Decoders", IEEE Trans. Commun., vol. 59, no. 12, pp. 3301-10, Dc 2011.

[2] http://www.3gpp.org/ftp/Specs/html-info/36212.htm, version 10.0.0.

[3] E. Boutillon, J.-L. Sanchez-Rojas, C. Marchand , "Compression of redundancy free trellis stages in Turbo-Decoder", Electronics Letters vol. 49, no. 7, pp. 460-462, Feb. 2013.

[4] L.Bahl, J.Cocke, F.Jelinek, and J.Raviv, "Optimal decoding of linear codes for minimizing symbol error rate", IEEE Trans. Inf. Theory, vol. IT-20(2), pp. 284-287, March 1974.

[5] E. Boutillon, C. Douillard, G. Montorsi, "Iterative decoding of concatenated convolutional codes: Implementation Issues", Proceedings of the IEEE, vol. 95, no.6, June 2007.

[6] A.J. Viterbi. "An intuitive justification and a simplified implementation of the MAP decoder for convolutional codes", IEEE J. Sel. Areas Commun., vol.16, pp. 260264, Feb. 1998.

[7] J. Dielissen, and J. Huisken, "State vector reduction for initialization of sliding window MAP", in Proc. 2nd Int. Symp. Turbo Codes, pp. 387-390, Sept. 2000.

[8] C. Weiss, C. Bettstetter, S. Riedel, D.J. Costello "Turbo decoding with tail-biting trellises", International Symposium on Signal, System and Electronics, pp. 343-348, Pisa, Italy, Sept. 1998.

[9] E. Boutillon, April 2014, http://www-labsticc.univ-ubs.fr/~boutillon/tc_agglo/tc_agglo.html.