A low-complexity dual trellis decoding algorithm for high-rate convolutional codes

Vinh Hoang Son Le, Charbel Abdel Nour, Catherine Douillard, Emmanuel Boutillon

A Low-Complexity Dual Trellis Decoding Algorithm for High-Rate Convolutional Codes

Vinh Hoang Son Le*, Charbel Abdel Nour*, Catherine Douillard* and Emmanuel Boutillon†
*IMT Atlantique, Lab-STICC, UMR CNRS 6285, F-29238 Brest, France
Email: {vinh.le, charbel.abdelnour, catherine.douillard}@imt-atlantique.fr
†Université Bretagne Sud, Lab-STICC, UMR CNRS 6285, F-56321 Lorient, France
Email: emmanuel.boutillon@univ-ubs.fr

***Abstract*—Decoding using the dual trellis is considered as a potential technique to increase the throughput of soft-input soft-output decoders for high coding rate convolutional codes. However, the dual Log-MAP algorithm suffers from a high decoding complexity. More specifically, the source of complexity comes from the soft-output unit, which has to handle a high number of extrinsic values in parallel. In this paper, we present a new low-complexity sub-optimal decoding algorithm using the dual trellis, namely the dual Max-Log-MAP algorithm, suited for high coding rate convolutional codes. A complexity analysis and simulation results are provided to compare the dual Max-Log-MAP and the dual Log-MAP algorithms. Despite a minor loss of about 0.2 dB in performance, the dual Max-Log-MAP algorithm significantly reduces the decoder complexity and makes it a first-choice algorithm for high-throughput high-rate decoding of convolutional and turbo codes.**

***Index Terms*—Convolutional codes, high coding rate, dual trellis, high-throughput decoder, low-complexity decoder, turbo codes**

I. Introduction

Nowadays, as the bandwidth becomes scarcer, many communication systems have to adopt error correction schemes with high coding rates in order to limit the occupied bandwidth. For convolutional codes, high coding rates $r = k/n$ (e.g. $r > 1/2$) are usually achieved by applying a puncturing pattern to a low-rate mother code so as to reduce the amount of transmitted bits [1]. When these codes are used as component codes in a turbo code [2], a soft-intput soft-output (SISO) algorithm is needed for iterative decoding. The symbol-by-symbol *maximum a posteriori* (MAP) algorithm [3] or its sub-optimal version in the logarithmic domain (Max-Log-MAP) [4] are usually employed to compute the soft output as well as the extrinsic information for the iterative process. However, decoding a high-rate punctured convolutional code using the MAP algorithm presents some drawbacks. First of all, whatever the coding rate, the decoder always employs the trellis of the low-rate non-punctured mother code. Therefore, the decoding throughput is limited by the decoding throughput of the mother code. Furthermore, the acquisition stages required to initialize the state metrics in the decoder become longer when the rate increases, thus reducing the efficiency and the throughput of the decoder [5].

This work was partially funded by the EPIC project of the EU's Horizon 2020 research and innovation programme under grant agreement No. 760150.

An alternative to decode a high-rate punctured convolutional code involves using its dual code. As shown in [6], the codewords of the dual code can be represented as sequences in the trellis of the reciprocal dual convolutional code, referred to as *dual trellis* in this paper. Then, a symbol-by-symbol MAP algorithm can be employed on the dual trellis to compute the *a posteriori* and extrinsic log-likelihood ratios (LLR). We refer to this decoding algorithm and to its derivation in the logarithmic domain as the *dual MAP* and *dual Log-MAP* algorithms, respectively. For codes with rates r greater than $1/2$, it is expected that decoding on the trellis of the rate-$(n-k)/n$ dual code yields higher throughput and lower decoding latency than decoding on the original trellis, since the dual trellis is shorter than the original one. In [7], two hardware implementations of a dual Log-MAP decoder and a conventional radix-4 Max-Log-MAP decoder are reported and compared in terms of throughput and circuit area, for the rate-$k/(k+1)$ constituent convolutional codes of the LTE standard ($k = 2, 4, 8, 16$). As expected, the decoding throughput of the dual Log-MAP decoder increases with the coding rate, while the decoding throughput of the Max-Log-MAP decoder remains unchanged. The price to pay for higher throughput is a larger circuit area of the dual Log-MAP decoder for all considered coding rates, since the dual Log-MAP algorithm has to process a large number of systematic and extrinsic information values in parallel. This area increase is currently regarded as the main obstacle to the extensive implementation of the dual Log-MAP algorithm.

In this paper, we propose a new decoding algorithm based on the dual trellis, called *dual Max-Log-MAP*. Compared to the dual Log-MAP algorithm, it exhibits a lower complexity while keeping the high-throughput property. More specifically, the number of look-up tables required in the soft output unit (SOU) is significantly reduced, compared to the dual Log-MAP algorithm. This reduction results in a decrease of the decoder circuit area at the cost of a minor penalty in performance compared to the dual Log-MAP algorithm.

The paper is organized as follows. Section II presents some basic concepts on the construction of the dual trellis, describes the dual MAP and dual Log-MAP decoding algorithms, and provides some considerations on the hardware implementation of the latter. Then, the main contribution of the paper, i.e. the proposed dual Max-Log-MAP algorithm, is described in

Section III. A computational complexity analysis is presented in Section IV as well as simulations results to assess the error correction performance of the proposed algorithm. Finally, Section V concludes the paper.

II. Decoding with the Dual Trellis

A. Dual Trellis Construction for High-Rate Punctured Convolutional Codes

The construction of the dual trellis is the first step towards decoding the dual code. The dual trellis construction procedure is based solely on the generator matrix of the mother code and on the puncturing pattern. Therefore, the dual trellis is known to the decoder prior to the decoding process but, contrary to the original code trellis, it is specific to a given coding rate and puncturing pattern.

Given a punctured convolutional code with coding rate $r = k/n$, there exists an associated dual code with coding rate $r^{\perp} = (n-k)/n$, generated by the reciprocal parity-check matrix $\widetilde{\mathbf{H}}(D)$ [6]. Hence, the dual trellis can be obtained by finding the matrix $\widetilde{\mathbf{H}}(D)$. To this end, a method is described in [8] for the specific case where the coding rate of the original punctured code is $r = k/(k+1)$. More recently, a generalized method was proposed in [9], allowing the dual trellis to be constructed for any coding rate $r = k/n$. The dual MAP algorithm or its logarithmic version dual Log-MAP can then be employed with the dual trellis to perform the decoding process.

B. The Dual MAP Algorithm

In this section, we assume that a binary information frame of size K is encoded by a rate-k/n punctured convolutional code yielding codewords of size $N = K \times n/k$, denoted by $\mathbf{c} = [c_1, \ldots, c_N]$. Each codeword is modulated using a binary phase-shift keying (BPSK) modulation where the output signal is "+1" if bit $c_j = 0$ or "−1" if bit $c_j = 1$. The modulated signal is then transmitted over the additive white Gaussian noise (AWGN) channel. At the receiver, let $\mathbf{y} = [y_1, \ldots, y_N]$ denote the received signal. The channel LLRs $\mathbf{L}^c = [L_1^c, \ldots, L_N^c]$ are then expressed as:

$$L_j^c = \log \frac{P\{y_j | c_j = 0\}}{P\{y_j | c_j = 1\}} = \frac{2}{\sigma^2} y_j. \tag{1}$$

where σ^2 is the noise variance.

If the decoder of the convolutional code takes part in an iterative process, then some *a priori* information has to be taken into account in the overall decoding process. For a conventional turbo code, the LLRs at the input of the decoder, denoted by $\mathbf{L}^I = [L_1^I, \ldots, L_N^I]$ are defined as follows:

$$L_j^I = \begin{cases} L_j^c + L_j^a, & \text{if } c_j \text{ is an information (systematic) bit,} \\ L_j^c, & \text{if } c_j \text{ is a parity bit,} \end{cases} \tag{2}$$

where L_j^a denotes the *a priori* LLR related to bit c_j.

At the output of the decoder, the *extrinsic* LLRs, denoted by $\mathbf{L^e} = \{L_1^e, \ldots, L_N^e\}$, can be obtained using either the original or dual trellis. In the latter case [6], [10], the input LLRs are first converted into bit metrics through the following hyperbolic tangent computation:

$$d_j = \tanh(L_j^I/2). \tag{3}$$

Then, the MAP algorithm is carried out using the dual trellis, performing the following steps: *branch metric calculation*, *forward recursion*, *backward recursion*, and finally, *extrinsic information calculation*.

Assuming a dual trellis constructed according to Section II-A for a given punctured convolutional code with rate $r = k/n$, we focus on section t of this trellis. The bit metrics related to this trellis section are $\{d_{nt+1}, \ldots, d_{nt+n}\} \in \mathbb{R}^n$. We denote by (s_t, s_{t+1}) a pair of states between time t and time $(t+1)$ having a transition in the trellis section. Furthermore, let $\{b_{nt+1}, \ldots b_{nt+n}\} \in \{0,1\}^n$ be the hard decisions carried by state transition (s_t, s_{t+1}). The dual MAP algorithm is implemented as follows [6], [10]:

- Branch metric calculation γ_t

$$\gamma_t(s_t, s_{t+1}) = \prod_{j=nt+1}^{nt+n} (d_j)^{b_j} \tag{4}$$

- Forward α_{t+1} and backward β_t metric recursions

$$\begin{cases} \alpha_{t+1}(s_{t+1}) = \sum\limits_{s_t} \alpha_t(s_t)\gamma_t(s_t, s_{t+1}) \\ \beta_t(s_t) = \sum\limits_{s_{t+1}} \beta_{t+1}(s_{t+1})\gamma_t(s_t, s_{t+1}) \end{cases} \tag{5}$$

- The *a posteriori* likelihood ratio (LR) z_j of each bit at position j, where $j = nt + i,\ i \in \{1, \ldots, n\}$, is then calculated in the dual code domain as

$$z_j = \frac{q_j^0}{q_j^1} = \frac{\sum\limits_{(s_t, s_{t+1})|b_j=0} \alpha_t(s_t)\gamma_t(s_t, s_{t+1})\beta_{t+1}(s_{t+1})}{\sum\limits_{(s_t, s_{t+1})|b_j=1} \alpha_t(s_t)\gamma_t(s_t, s_{t+1})\beta_{t+1}(s_{t+1})}. \tag{6}$$

- The resulting extrinsic information, denoted by u_j, is then

$$u_j = z_j/d_j, \quad \text{where } j = nt + i,\ i \in \{1, \ldots, n\}. \tag{7}$$

- Finally, the extrinsic information for the original code at position j, denoted by L_j^e, is obtained as [6]

$$L_j^e = \log \frac{1 + u_j}{1 - u_j}. \tag{8}$$

Just like the conventional MAP algorithm, the dual MAP algorithm requires a large number of multiplications, divisions and additions and is therefore too complex for practical hardware implementations. Consequently, to make decoding on the dual trellis feasible, the application of the dual MAP algorithm in the logarithmic domain was investigated in [11].

C. The Dual Log-MAP Algorithm with Sign-Magnitude Representation

As shown in (3), the input bit metric of the dual MAP decoder can be positive or negative. Therefore, to represent it in the logarithmic domain, one can resort to the sign-magnitude

(SM) representation as in [11]. With this representation, a real number x is represented as

$$x = (-1)^{X_s} \exp(-X_m) \triangleq [X_s; X_m], \tag{9}$$

where $X_s \in \{0; 1\}$ and $X_m = -\log(|x|)$ are the sign and the magnitude of x, respectively. Then, the arithmetic operations involved in the decoding algorithm can be expressed as follows, using the SM representation:

- Sign-magnitude multiplication (SMM):

$$xy \triangleq [X_s \oplus Y_s; X_m + Y_m]. \tag{10}$$

- Sign-magnitude division (SMD):

$$x/y \triangleq [X_s \oplus Y_s; X_m - Y_m]. \tag{11}$$

- Sign-magnitude addition (SMA):

$$x + y \triangleq \begin{cases} \big[X_s; X_m - \log\big(1+(-1)^{X_s+Y_s}\exp(Y_m - X_m)\big)\big], \\ \quad \text{if } Y_m > X_m, \\ \big[Y_s; Y_m - \log\big(1+(-1)^{X_s+Y_s}\exp(X_m - Y_m)\big)\big], \\ \quad \text{otherwise.} \end{cases} \tag{12}$$

With this representation, the multiplication and division operators used by the dual MAP decoder are replaced by SMM and SMD operators, mainly consisting in adders, in the dual Log-MAP (dual LM) decoder. However, the SMA operator in (12) requires the implementation of look-up tables (LUT) to perform the two following functions:

$$f(\Delta) = \log\big(1 + \exp(-\Delta)\big), \tag{13}$$
$$g(\Delta) = \log\big(1 - \exp(-\Delta)\big), \tag{14}$$

where Δ is a positive real number. The hardware architecture of the SMA is shown in Fig. 1. In the original trellis, only function $f(\Delta)$ is employed and can be replaced in practice by a scaling factor in the Max-Log-MAP decoder [12]. Differently, the bit metrics in the dual MAP algorithm can be negative, thus requiring the use of function $g(\Delta)$. Function $g(\Delta)$ is not bounded when Δ goes to zero, therefore, it can not be discarded or approximated as in the case of function $f(\Delta)$ in the Max-Log-MAP or Log-MAP decoder.

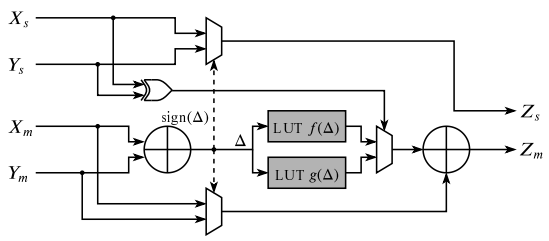

Fig. 1. Hardware architecture of an SMA operator performing $z = x + y$.

Finally, the extrinsic LLR in (8) can be expressed using the SM representation as [7]:

$$L_j^e = \begin{cases} -\log\Big(\big|\tanh(U_{j,m}/2)\big|\Big), & \text{if } U_{j,s} = 0, \\ \log\Big(\big|\tanh(U_{j,m}/2)\big|\Big), & \text{otherwise,} \end{cases} \tag{15}$$

where $U_{j,s}$ and $U_{j,m}$ are the sign and the magnitude of u_j, respectively.

D. Drawbacks of the Dual LM Algorithm for Hardware Implementation

The required use of two LUTs to implement an SMA operator has a penalizing impact on the dual LM decoder implementation. In order to achieve the highest possible throughput, the soft-output unit (SOU), in charge of computing the *a posteriori* LRs, has to perform simultaneously the computation of (6) for every information bit in a section of the dual trellis. The corresponding architecture is shown in Fig. 2 for a radix-2 dual trellis with 4 states. Furthermore, the number of required SMA operators employed increases with the number of states and with the rate of the original convolutional code: if the code has 2^ν states and rate $r = k/n$, the number of SMA operators required for the SOU of the dual LM decoder is $2k \times (2^\nu - 1)$. Since each SMA operator requires two LUTs, the SOU consists of a total of $4k \times (2^\nu - 1)$ LUTs. Consequently, the implementation of the dual LM decoder is only of interest for very high coding rates: it was reported in [7] that, for a rate-$8/9$ turbo code, the SOUs occupy more than 30 % the circuit area of the turbo decoder.

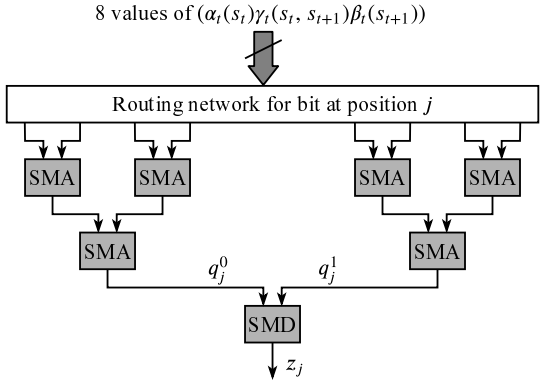

Fig. 2. Hardware architecture for the calculation of (6) for bit at position j.

In what follows, we propose a new sub-optimal decoding algorithm to mitigate the complexity issue of the dual LM algorithm [7], [11]. This algorithm is referred to as the dual Max-Log-MAP (dual MLM) algorithm in the rest of the paper.

III. Proposed Sub-Optimal Low-Complexity Decoding Algorithm

A. Max-Log Approximation for the Extrinsic Information Calculation

The complexity of the dual LM decoder can be lowered by limiting the use of LUTs to compute (13) and (14). These LUTs are mainly used to derive the soft outputs according to

(6) in the SOUs. Therefore, we consider approximating the calculation in the logarithmic domain of q_j^0 and q_j^1 for each bit at position j in the trellis section, where:

$$q_j^0 = \sum_{(s_t,s_{t+1})|b_j=0} \alpha_t(s_t)\gamma_t(s_t,s_{t+1})\beta_{t+1}(s_{t+1}), \quad (16)$$

$$q_j^1 = \sum_{(s_t,s_{t+1})|b_j=1} \alpha_t(s_t)\gamma_t(s_t,s_{t+1})\beta_{t+1}(s_{t+1}). \quad (17)$$

For the calculation of q_j^0, if $\mathcal{N}$ branches have bit decision equal to zero at position j, we denote by $\mathcal{S}_j = \{1,\ldots,\mathcal{N}\}$ the index set of those branches and by w_s the metric of branch $s \in \mathcal{S}_j$ corresponding to state transition (s_t, s_{t+1}):

$$w_s = \alpha_t(s_t)\gamma_t(s_t,s_{t+1})\beta_{t+1}(s_{t+1}). \quad (18)$$

Then, (16) can be rewritten as:

$$q_j^0 = \sum_{s\in\mathcal{S}_j} w_s. \quad (19)$$

The value of w_s can be negative or positive and the corresponding sign is not known *a priori*. Therefore, with the SM representation, one cannot apply the max-log approximation

$$\log\big(\sum_{i=1}^{N} e^{x_i}\big) \approx \max_{i=1,\ldots,N}\{x_i\},$$

to the summation in (19). However, if we denote by $\mathcal{S}_j^+$ and $\mathcal{S}_j^-$ the index sets of branches having positive metrics and negative metrics, respectively, (19) can be expressed as

$$q_j^0 = \Big(\sum_{r\in\mathcal{S}_j^+} w_r\Big) + \Big(\sum_{p\in\mathcal{S}_j^-} w_p\Big). \quad (20)$$

Since the elements in the two summations terms in (20) have the same sign value, these summations can be expressed in SM representation as

$$\sum_{r\in\mathcal{S}_j^+} w_r = \Big[0; -\log\big(\sum_{r\in\mathcal{S}_j^+} e^{-W_{r,m}}\big)\Big], \quad (21)$$

$$\sum_{p\in\mathcal{S}_j^-} w_p = \Big[1; -\log\big(\sum_{r\in\mathcal{S}_j^-} e^{-W_{p,m}}\big)\Big]. \quad (22)$$

where $W_{r,m}$ and $W_{p,m}$ are the magnitudes of w_r and w_p, respectively. Then, based on the max-log approximation, (21) and (22) can be approximated in the SM domain as

$$\sum_{r\in\mathcal{S}_j^+} w_r \approx \big[0; -\max_{r\in\mathcal{S}_j^+}(-W_{r,m})\big] = \big[0; \min_{r\in\mathcal{S}_j^+}(W_{r,m})\big], \quad (23)$$

$$\sum_{p\in\mathcal{S}_j^-} w_p \approx \big[1; -\max_{p\in\mathcal{S}_j^-}(-W_{p,m})\big] = \big[1; \min_{p\in\mathcal{S}_j^-}(W_{p,m})\big]. \quad (24)$$

Finally, replacing both terms in (20) by (23) and (24), the SM representation of q_j^0 after the SMA operation can be written:

$$Q_{j,s}^0 = W_{l_j,s}, \quad (25)$$

$$Q_{j,m}^0 \approx W_{l_j,m} - \log(1 - e^{-\Delta_j}), \quad (26)$$

where $l_j = \arg\min_{i\in\mathcal{S}_j}\{W_{i,m}\}$ and $\Delta_j = |\min_{r\in\mathcal{S}_j^+}\{W_{r,m}\} - \min_{p\in\mathcal{S}_j^-}\{W_{p,m}\}|$. We can observe that the calculation of (26) involves only one LUT for function $g(\Delta_j)$, regardless of the number of considered branches. Consequently, through branch rearrangement, we can significantly reduce the number of LUTs employed by the dual MAP decoder.

However, another problem arises when implementing (26). Intuitively, (26) can be computed by finding the minimum metric in $\mathcal{S}_j^+$ and the minimum metric in $\mathcal{S}_j^-$. Then, Δ_j and $W_{l_j,s}$ are the results of a subtraction and a sign detection, respectively. However, the number of branches in $\mathcal{S}_j^+$ and $\mathcal{S}_j^-$ is not constant. It depends on the considered dual trellis section and varies with each received frame. This variability requires the hardware implementation to be able to handle all possible sizes for $\mathcal{S}_j^+$ and $\mathcal{S}_j^-$, which comes at a cost in complexity. However, this problem can be easily solved by resorting to a recently proposed method, namely the local soft-output Viterbi algorithm (local SOVA) [13], to compute $W_{l_j,s}$, $W_{l_j,m}$ and Δ_j in (25) and (26).

B. The Local Soft-Output Viterbi Algorithm

The algorithm presented in this section is a variant of the original one recently proposed in [13]. First, we present the algorithm for the general case and, then, its application to the computation of (26) is derived.

The local SOVA performs operations on 3-tuple entities, called *paths*. A path P consists of a metric value denoted by M, a sign value S, and a reliability value related to S, denoted by R:

$$P = \{M, S, R\} \in \mathbb{R} \times \{0,1\} \times \mathbb{R}^+, \quad (27)$$

where $\mathbb{R}$ is the set of real numbers and $\mathbb{R}^+$ is the set of positive real numbers.

Given a set of $\mathcal{N}$ paths, each has a sign value which is positive ($S = 0$) or negative ($S = 1$). Also, each path has a pre-computed path metric M and its initial reliability value R is set to $+\infty$ or to the largest possible value achievable with quantization. Then, the local SOVA proceeds as shown in Algorithm 1. The outcome of the algorithm is the path with the minimum metric value among all paths. The corresponding sign and metric are provided as well as the associated reliability value, which is the minimum metric among the set of competing paths having a different sign value.

Algorithm 1 can be directly used to calculate (25) and (26). The number of paths $\mathcal{N}$ to be considered is the cardinality of set $\mathcal{S}_j$. The metric of path $p \in \mathcal{S}_j$ is w_p, which is written $[W_{p,s}, W_{p,m}]$ using the SM representation. The $\mathcal{N}$ paths are initialized with:

$$P_p = \{M_p, S_p, R_p\} = \{W_{p,m}, W_{p,s}, +\infty\}, p = 1\cdots\mathcal{N}. \quad (28)$$

Algorithm 1 The local SOVA

Initialization: $\mathcal{N}$ paths $\{P_1, \ldots, P_{\mathcal{N}}\}$;
$P_i = \{M_i, S_i, R_i\}$, for $i = 1, \ldots, \mathcal{N}$;
$\mathcal{L} = \log_2(\mathcal{N})$ as the number of layers;
for each layer $l = 1, \ldots, \mathcal{L}$ **do**
  for each path $p = 1, \ldots, 2^{(\mathcal{L}-l)}$ **do**
    $a = \arg\min_{j \in \{2p-1, 2p\}} \{M_j\}$;
    $b = \arg\max_{j \in \{2p-1, 2p\}} \{M_j\}$;
    if $S_a = S_b$ **then**
      $R_p = \min(R_a, R_b)$;
    else
      $R_p = \min(R_a, M_b)$;
    end if
    $M_p = M_a$; $S_p = S_a$;
  end for
end for
Output:
M_1: minimum metric among all paths,
S_1: sign value of the path with minimum metric,
R_1: minimum metric among paths having sign value different from S_1.

After having run the local SOVA according to Algorithm 1, the output path is $\{M_1, S_1, R_1\}$. If $l_j = \arg\min_{p \in \mathcal{S}_j} \{W_{p,m}\}$, and if we assume that $l_j \in \mathcal{S}_j^+$, then we have:

$$\begin{cases} M_1 = W_{l_j,m} = \min\limits_{r \in \mathcal{S}_j^+} \{W_{r,m}\}, & (29) \\ S_1 = W_{l_j,s}, & (30) \\ R_1 = \min\limits_{p \in \mathcal{S}_j^-} \{W_{p,m}\}. & (31) \end{cases}$$

If $l_j \in \mathcal{S}_j^-$, $\mathcal{S}_j^+$ and $\mathcal{S}_j^-$ have to be swapped in (29) and (31). The value of Δ_j in (26) is then

$$\Delta_j = |\min_{r \in \mathcal{S}_j^+} \{W_{r,m}\} - \min_{p \in \mathcal{S}_j^-} \{W_{p,m}\}| = |M_1 - R_1|. \quad (32)$$

To summarize, the computation of q_j^0 using the local SOVA results in:

$$\begin{cases} Q_{j,s}^0 = S_1 & (33) \\ Q_{j,m}^0 = M_1 - \log\left(1 - e^{-|M_1 - R_1|}\right) & (34) \end{cases}$$

In addition, as presented in [13], the practical implementation of the local SOVA can be carried out in a dichotomous fashion using elementary operations, called *Merge*, that each processes two paths and are organized in layers. In Algorithm 1, two paths P_{2p-1} and P_{2p} at layer $l-1$ are merged into a resulting path P_p at layer l. The overall local SOVA decoder can therefore be implemented as a tree structure composed of elementary Merge operators. For example, the overall structure of the local SOVA decoder to calculate (25) and (26) with $\mathcal{N} = 4$ paths is described in Fig. 3 in the form of a binary tree with $\mathcal{L} = \log_2(\mathcal{N}) = 2$ layers. A possible architecture for the elementary Merge operator is described in Fig. 4.

Therefore, the local SOVA decoder makes it possible to use a predefined hardware structure to compute the simplified soft output of a dual LM decoder.

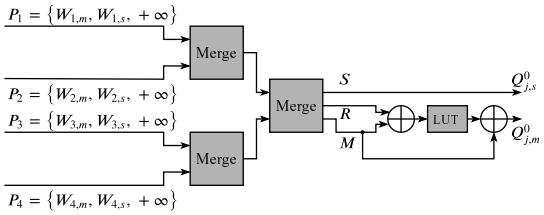

Fig. 3. Proposed hardware architecture of a local SOVA decoder used to compute (25) and (26) for $\mathcal{N} = 4$.

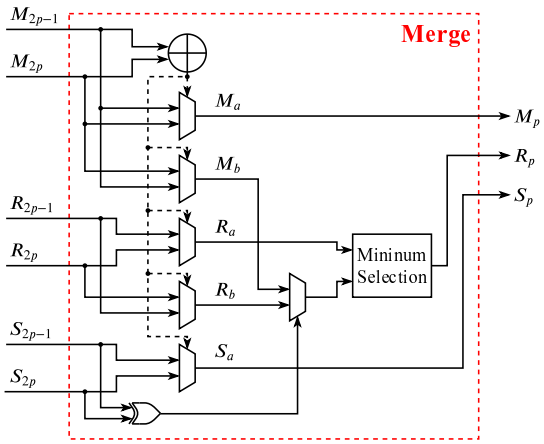

Fig. 4. Hardware architecture of a Merge operator that selects the minimum metric with its corresponding sign and updates the reliability value.

C. The Dual Max-Log-MAP Algorithm

Based on the above results, we propose a new a low-complexity algorithm, called dual Max-Log-MAP (dual MLM), to decode high coding rates convolutional codes using theit dual trellis.

The decoding procedure is described as follows. Note that we skip the branch metric calculation since it is not different from the dual LM algorithm.

1) Forward or Backward Metric Recursion: As expressed in (5), the metric recursion involves summations in the probability domain. The number of required additions depends on the dual trellis structure. If a radix-2^T dual trellis (2^T state transitions enter and leave each state) is considered, the decoder needs $(2^T - 1)$ adders for each backward or forward metric recursion. With the SM representation, two LUTs are required for each SMA operator. Hence, a total number of $2\times(2^T - 1)$ LUTs are necessary for a straightforward implementation of each recursion.

However, thanks to the proposed SMA approximation and using the the local SOVA for its implementation, the number of LUTs needed to implement each recursion (5) using the SM representation is decreased to one.

2) Dual Extrinsic and A Posteriori Information Calculation: The extrinsic information calculation for each bit in a dual trellis section is given by (6). It involves the calculation of q_j^0 and q_j^1 for each position j as expressed in (16) and (17), respectively. The corresponding calculation has already been dealt with in Sections III-A and III-B. Next, the *a posteriori*

information z_j can be easily derived using an SMD operator to compute (7).

3) Scaling Factors for the Extrinsic Information Conversion: As a final step, the conversion of the extrinsic information from the dual trellis domain to the original trellis domain (15) is necessary. However, due to the above-mentioned approximations, the resulting extrinsic information values in the dual trellis domain are found to be over-estimated, which in turn produce under-estimated values in the original trellis domain. Therefore, we employed scaling factors to mitigate this over-estimation problem, a technique commonly used for sub-optimal SISO decoding algorithms such as the Max-Log-MAP [12] or SOVA [14] algorithms. Therefore, we employed two *scaling factors*, denoted by ϕ_1 and ϕ_2, for the conversion of the extrinsic information into the original trellis domain:

$$L_j^e = \begin{cases} -\phi_1 \log\Big(\big|\tanh(\phi_2 \frac{U_{j,m}}{2})\big|\Big), & \text{if } U_{j,s} = 1, \\ \phi_1 \log\Big(\big|\tanh(\phi_2 \frac{U_{j,m}}{2})\big|\Big), & \text{otherwise.} \end{cases} \quad (35)$$

In practice, we performed a computer search in an effort to find the values of ϕ_1 and ϕ_2 providing the best error correcting performance. These values are identical for all decoding iterations.

4) Latency and Throughput: Given a punctured convolutional code with a high coding rate of $r = k/n$, let K and N denote the information size and its corresponding codeword size, respectively. From the algorithmic perspective, the trellis of the dual code has $N-K$ trellis steps while the trellis of the original code has K trellis steps. Therefore, if $K > N - K$ and if it takes one clock cycle to decode each trellis step, the dual MLM decoder will have a lower latency than the classical MLM decoder due to its lower number of trellis steps to decode. In terms of throughput, the dual MLM can produce k soft outputs per trellis section. Hence, in principal, the throughput of the decoder with the dual MLM can potentially be increased by a factor of k compared to the classical MLM that produces one soft output per trellis step. Nevertheless, the actual latency and throughput of a decoder depend on the chosen architecture that might employ different levels of parallelism techniques to the classical MLM or the dual MLM decoder depending on the supported implementation constraints such as are, throughput and latency.

IV. Complexity analysis and simulation results

In order to illustrate the benefits of the proposed dual MLM algorithm, we first perform an analysis of the complexity savings due to the simplifications and then present simulations results to assess its error correction performance.

First, a simplified complexity comparison, in terms of number of adders and LUTs required by the decoder, is carried out between the dual LM and the proposed dual MLM algorithms, for an 8-state (3 memory elements) convolutional code with various coding rates $k/(k+1)$. In this comparison, we exclude the calculation of the branch metrics since it is the same for both algorithms. As shown in Table I, despite a minor increase in the number of adders, the dual MLM algorithm uses significantly less LUTs than the dual ML algorithm. More specifically, due to the max-log approximation, the use of function $f(\Delta)$ is no longer needed in (13) and the use of function $g(\Delta)$ is also limited, thanks to the local SOVA. As the coding rate increases, the percentage of LUTs saved in the dual MLM decoder also increases. For instance, with coding rate $4/5$, the total number of LUTs used in the dual LM decoder is 144, compared to 24 LUTs in the dual MLM decoder. When raising the coding rate up to $8/9$, dual MLM decoding requires only 32 LUTs compared to 256 for dual LM decoding.

TABLE I
Complexity comparison between the dual Log-MAP and the dual Max-Log-MAP decoders for various coding rates

Coding rate	dual Log-MAP			dual Max-Log-MAP		
	Adders	LUTs		Adders	LUTs	
		$\mathbf{f(\Delta)}$	$\mathbf{g(\Delta)}$		$\mathbf{f(\Delta)}$	$\mathbf{g(\Delta)}$
4/5	168	72	72	184	0	24
8/9	288	128	128	320	0	32
16/17	528	240	240	592	0	48

Next, we conducted several numerical simulations to assess and compare the performance of the dual LM and the dual-MLM algorithms. For validation purposes, we also included the performance of the Max-Log-MAP (MLM) decoder on the original trellis as a reference.

In the simulations, information frames are encoded by the turbo code consisting of two identical LTE constituent recursive systematic convolutional (RSC) codes [15] with generator matrix

$$\mathbf{G_{LTE}}(D) = \begin{pmatrix} 1 & \frac{1+D+D^3}{1+D^2+D^3} \end{pmatrix}. \quad (36)$$

For the internal interleaver, we chose an almost regular permutation (ARP) interleaver [16] defined by

$$\pi(i) = \Big(Pi + S\big(i \bmod Q\big)\Big) \bmod K, \quad i = 1, \ldots, K. \quad (37)$$

The parameters of the ARP interleaver are given in Table II. Furthermore, puncturing was employed to achieve different high coding rates, and the puncturing patterns of the parity bits were jointly optimized with the ARP interleaver as described in [17]. Table III shows the puncturing patterns used in the simulations for the coding rates $4/5$, $8/9$ and $16/17$ of the constituent convolutional code.

TABLE II
ARP interleaver parameters

Q	P	$\Big(S(0), \ldots, S(Q-1)\Big)$
16	383	(8, 80, 311, 394, 58, 55, 250, 298, 56, 197, 280, 40, 229, 40, 136, 192)

The encoded frames are modulated using a BPSK modulation and are sent through an additive white Gaussian noise (AWGN) channel. All three decoding algorithms use a fixed-point representation of data. The influence of channel data quantization on performance was observed to be similar for the three decoders. The simulations were carried out with channel

TABLE III
PARITY PUNCTURING PATTERNS FOR VARIOUS CODING RATES

Turbo rate	CC rate	Parity puncturing pattern
2/3	4/5	1000
4/5	8/9	01000000
8/9	16/17	0100000000000000

values quantized on 6 bits. The number of iterations is set to 8 for all decoders. For the dual MLM decoder, the values of the scaling factors (ϕ_1, ϕ_2) are $(1.3, 0.75)$ for $r = 2/3$, and $(1.15, 0.75)$ for $r = 4/5$ and $r = 8/9$. The simulation results are shown in Fig. 5 and Fig. 6 for information frame sizes $K = 400$ bits and $K = 992$ bits, respectively. We can see that the dual LM and the MLM algorithms yield similar error rate, as expected. However, the dual MLM algorithm entails a loss of about $0.2 - 0.3$ dB compared to the dual LM algorithm at coding rate $r = 2/3$ but this loss reduces to $0.1 - 0.2$ dB at coding rates $r = 4/5$ and $r = 8/9$. Therefore, the dual MLM algorithm can be regarded as a sub-optimal but low-complexity decoding algorithm compared to the dual LM algorithm.

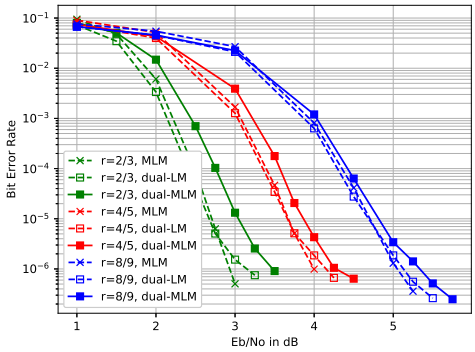

Fig. 5. Performance comparison between the Max-Log-MAP, the dual Log-MAP and the dual Max-Log-MAP algorithms with $K = 400$ bits.

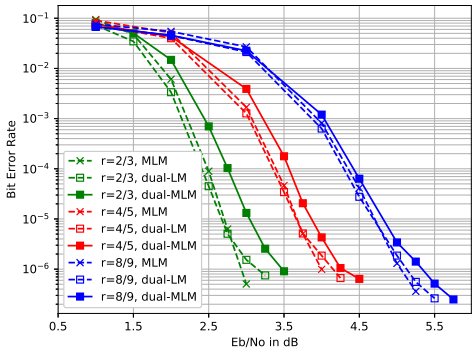

Fig. 6. Performance comparison between the Max-Log-MAP, the dual Log-MAP and the dual Max-Log-MAP algorithms with $K = 992$ bits.

V. CONCLUSIONS

In this paper, we proposed a new soft-input soft-output decoding algorithm using the dual trellis of high-rate punctured convolutional codes, namely the dual Max-Log-MAP algorithm. We showed that the state metric recursions and the extrinsic information calculations can then be reformulated using the max-log approximation and can be implemented with the local SOVA architecture. A complexity analysis was conducted, showing that the number of look-up tables employed in the decoder can then be considerably reduced compared to the dual Log-MAP algorithm. Also, based on numerical simulations, we observed that dual Max-Log-MAP decoding yields only a minor loss of about 0.2 dB in performance at 10^{-6} of bit error rate compared to dual Log-MAP decoding. Therefore, it can be considered as a viable and practical low-complexity sub-optimal decoding algorithm.

Moreover, as a decoding algorithm on the dual trellis, when high coding rates are considered, the dual Max-Log-MAP decoder inherits the high-throughput and low latency properties from the dual Log-MAP decoder [7]. This makes it a first-choice algorithm for high-throughput high-rate decoding of convolutional and turbo codes. Future work will focus on the hardware implementation of this algorithm.

REFERENCES

[1] J. Cain, G. Clark, and J. Geist, "Punctured convolutional codes of rate(n-1)/n and simplified maximum likelihood decoding," *IEEE Trans. Inf. Theory*, vol. 25, no. 1, pp. 97–100, January 1979.

[2] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes," in *Proc. IEEE Int. Conf. Commun.*, May 1993, pp. 1064–1070.

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

[4] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain," in *Proc. IEEE Int. Conf. Commun.*, vol. 2, June 1995, pp. 1009–1013 vol.2.

[5] E. Boutillon, C. Douillard, and G. Montorsi, "Iterative decoding of concatenated convolutional codes: Implementation issues," *Proc. IEEE*, vol. 95, no. 6, pp. 1201–1227, June 2007.

[6] S. Riedel, "MAP decoding of convolutional codes using reciprocal dual codes," *IEEE Trans. Inf. Theory*, vol. 44, no. 3, pp. 1176–1187, 1998.

[7] C. Lin, C. Wong, and H. Chang, "A 40 nm 535 Mbps multiple code-rate turbo decoder chip using reciprocal dual trellis," *IEEE J. Solid-State Circuits*, vol. 48, no. 11, pp. 2662–2670, Nov 2013.

[8] A. Graell i Amat, G. Montorsi, and S. Benedetto, "Design and decoding of optimal high-rate convolutional codes," *IEEE Trans. Inf. Theory*, vol. 50, no. 5, pp. 867–881, May 2004.

[9] V. Le, C. Abdel Nour, E. Boutillon, and C. Douillard, "Dual trellis construction for high-rate punctured convolutional codes," in *PIMRC Workshop*, Sep. 2019, pp. 1–7.

[10] S. Srinivasan and S. S. Pietrobon, "Decoding of high rate convolutional codes using the dual trellis," *IEEE Trans. Inf. Theory*, vol. 56, no. 1, pp. 273–295, Jan 2010.

[11] G. Montorsi and S. Benedetto, "An additive version of the SISO algorithm for the dual code," in *Proc. IEEE Int. Symp. Inf. Theory*, June 2001, p. 27.

[12] J. Vogt and A. Finger, "Improving the max-log-MAP turbo decoder," *Electronics Letters*, vol. 36, no. 23, pp. 1937–1939, Nov 2000.

[13] V. 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. Comm.*, 2020.

[14] Chuan Xiu Huang and A. Ghrayeb, "A simple remedy for the exaggerated extrinsic information produced by the SOVA algorithm," *IEEE Trans. Wireless Comm.*, vol. 5, no. 5, pp. 996–1002, May 2006.

[15] *LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and channel coding*, ETSI TS 136 212 V14.8.0, Jan 2019.

[16] C. Berrou, Y. Saouter, C. Douillard, S. Kerouedan, and M. Jezequel, "Designing good permutations for turbo codes: towards a single model," in *IEEE Int. Conf. Comm.*, vol. 1, June 2004, pp. 341–345.

[17] R. Garzón-Bohórquez, C. Abdel Nour, and C. Douillard, "Protograph-based interleavers for punctured turbo codes," *IEEE Trans. Commun.*, vol. 66, no. 5, pp. 1833–1844, May 2018.