Low-Complexity Belief Propagation Decoding by Approximations with Lookup-Tables

Thorsten Clevorn and Peter Vary

Institute of Communication Systems and Data Processing (ind)
Aachen University (RWTH), Germany
Email: {clevorn,vary}@ind.rwth-aachen.de

***Abstract*— Belief propagation decoding of low-density parity-check codes or one-step majority logic decodable codes has been proven to be a very powerful coding scheme. In this paper an approximation for the belief propagation algorithm, also known as sum-product decoding, is presented which uses correction functions, implemented as precomputed lookup-tables, to significantly reduce the computational complexity. The new lookup-sum algorithm requires no multiplications, divisions, exponential or logarithmic operations in the iterative process. Already for lookup-tables containing a single entry simulation results show that the performance of non-approximated belief propagation can be approached by 0.1 dB in E_b/N_0. With slightly larger tables a performance not noticeably differing from non-approximated belief propagation can be achieved.**

I. Introduction

Following the discovery of Turbo codes [1], [2], [3] and the resulting increased interest in iterative decoding algorithms, *low-density parity-check* (LDPC) codes, originally presented in [4], were rediscovered in [5], [6]. LDPC codes were shown to have a performance similarly to Turbo codes. The standard decoding algorithm for LDPC codes is *belief propagation* (BP) [7], [6], also known as *sum-product* decoding. Since both, LDPC codes and *one-step majority logic decodable* (OSMLD) codes, are based upon orthogonal parity-check sums for each bit, which can be written in form of a matrix, BP decoding is also applicable for the latter one [8]. It was also shown in [8] that for medium block sizes, *difference set cyclic* (DSC) codes [9], a sub-class of OSMLD codes, outperform their corresponding regular LDPC codes.

In this paper an approximation for the computationally complex BP algorithm is derived. The proposed algorithm uses correction functions, implemented by precomputed lookup-tables. Simulations with DSC codes and a channel with additive white Gaussian noise (AWGN) show that a performance very similar to the standard BP algorithm without approximations can be achieved.

II. Belief Propagation

In the following we will shortly review the BP algorithm to introduce some notations and show the connection to the *box-plus* operation defined in [3].

We consider a code with a $P\times N$ parity-check matrix $\underline{B}$ containing elements $B_{pn}\in\{0,1\}$, $p=1,\dots P$ and $n=1,\dots N$. The rows of $\underline{B}$ are the P parity-check vectors $\underline{b}_p$. For each column n of $\underline{B}$ there exists a sub-matrix $\underline{\mathcal{B}}_n$ of parity-check vectors $\underline{b}_p$ with $B_{pn}=1$. Thus, $\underline{\mathcal{B}}_n$ comprises all rows of $\underline{B}$ with a one in column n. Furthermore, no other column of $\underline{\mathcal{B}}_n$ has more than a single non-zero element. For example if we use

$$\underline{B}=\begin{pmatrix}1&1&0&1&0&0\\1&0&1&0&1&0\\0&1&1&0&0&1\\0&0&0&1&1&1\end{pmatrix}$$

we get

$$\underline{\mathcal{B}}_2=\begin{pmatrix}1&1&0&1&0&0\\0&1&1&0&0&1\end{pmatrix}.$$

We assume the transmission of the bipolar code word $\underline{y}=(y_1,\dots y_N)$, $y_n\in\{+1,-1\}$, and the reception of $\underline{z}=(z_1,\dots z_N)$, $z_n\in\mathbb{R}$, which is distorted by AWGN of zero mean and variance $N_0/2$.

When using log-likelihood ratios, or L-values [3], for BP decoding [8], the elements z_n of the received word $\underline{z}$ are converted to L-values $r_n^{(0)}:=(4/N_0)\cdot z_n$. The proposed algorithm uses only L-values. Next, a $P\times N$ matrix $\underline{Z}$ is initialized according to $\underline{B}$ as $Z_{pn}^{(0)}=r_n^{(0)}$ if $B_{pn}=1$, and $Z_{pn}^{(0)}=0$ otherwise.

The first step of each iteration i is the computation of a matrix $\underline{E}^{(i)}$ with $E_{pn}^{(i)}=0$ for $B_{pn}=0$, and

$$E_{pn}^{(i)}=2\,\mathrm{atanh}\left(\prod_{j\in\mathrm{supp}(\underline{b}_p)\setminus n}\tanh(Z_{pj}^{(i)}/2)\right)\quad(1)$$

for $B_{pn}=1$, with $\mathrm{supp}(\underline{b}_p):=\{j|B_{pj}=1\}$. Since the $Z_{pj}^{(i)}$ are L-values, (1) can be defined as *box-plus* operations $\boxplus$ according to [3] and abbreviated by

$$E_{pn}^{(i)}=\sum_{j\in\mathrm{supp}(\underline{b}_p)\setminus n}\boxplus Z_{pj}^{(i)}.\quad(2)$$

In (1) and (2) always just one element is excluded from the respective row, $j\in\mathrm{supp}(\underline{b}_p)\setminus n$. Thus, it is obviously more efficient, at least for rows with a not too small weight, to first compute the result for the complete row, $j\in\mathrm{supp}(\underline{b}_p)$, and afterwards exclude

the particular element $j=n$. Therefore, we divide the computation of $E_{pn}^{(i)}$ into two parts:

- *Part 1* contains the computation of a vector $\underline{s}^{(i)}$ with values for the complete rows by

$$s_p^{(i)} = \prod_{j \in \mathrm{supp}(\underline{b}_p)} \tanh(Z_{pj}^{(i)}/2). \quad (3)$$

- *Part 2* consists of the extraction of single $E_{pn}^{(i)}$ by

$$E_{pn}^{(i)} = 2\,\mathrm{atanh}\left(\frac{s_p^{(i)}}{\tanh(Z_{pn}^{(i)}/2)}\right). \quad (4)$$

In the next step of each iteration the obtained *extrinsic* information $E_{pn}^{(i)}$ is summed for each n

$$e_n^{(i)} = \sum_{p,\, \underline{b}_p \in \underline{\mathcal{B}}_n} E_{pn}^{(i)} \quad (5)$$

and then added to $r_n^{(0)}$

$$r_n^{(i+1)} = r_n^{(0)} + e_n^{(i)}. \quad (6)$$

Afterwards a component-wise hard decision is made $\hat{\underline{y}}^{(i+1)} = \mathrm{sgn}(\underline{r}^{(i+1)})$, $\hat{\underline{y}}^{(i+1)} \in GF(2)$. If the maximum number of iterations is reached or if $\hat{\underline{y}}^{(i+1)}$ fulfills the terminating condition $(\underline{B}\cdot\hat{\underline{y}}^{(i+1)}) \bmod 2 = \underline{0}$, the iterative process is stopped and $\hat{\underline{y}}^{(i+1)}$ serves as the decoding output.

Otherwise the elements Z_{pn} with $B_{pn} = 1$ are updated in the last step by

$$\begin{aligned} Z_{pn}^{(i+1)} &= Z_{pn}^{(0)} + \sum_{l,\, \underline{b}_l \in \underline{\mathcal{B}}_n \setminus \underline{b}_p} E_{ln}^{(i)} \\ &= Z_{pn}^{(0)} + e_n^{(i)} - E_{pn}^{(i)} \end{aligned} \quad (7)$$

and after increasing the iteration index, $i = i+1$, the next iteration starts with (1). In (7) $E_{pn}^{(i)}$ is excluded from the summation for the new $Z_{pn}^{(i+1)}$ to mitigate error propagation.

III. Proposed Approximations

In this section we derive approximations using lookup-tables for a high computational efficiency for part 1, (3), and part 2, (4), of the first step of each iteration in BP decoding. To simplify the notation we will consider L-values $L_j = Z_{pj}^{(i)}/2$ in general. Since only L-values shall be used, (3) is modified to

$$\bar{s}_p^{(i)} = 2\,\mathrm{atanh}\left(\prod_{j \in \mathrm{supp}(\underline{b}_p)} \tanh(Z_{pj}^{(i)}/2)\right), \quad (8)$$

which yields in *box-plus* arithmetic

$$2\,\mathrm{atanh}\left(\prod_{j \in \mathrm{supp}(\underline{b}_p)} \tanh(Z_{pj}^{(i)}/2)\right) = \sum_{j=1}^{J} \boxplus L_j \triangleq L_\Sigma, \quad (9)$$

assuming $\mathrm{supp}(\underline{b}_p)$ has J elements. The *box-plus* operation for two L-values L_1 and L_2 is defined as [3]

$$L_1 \boxplus L_2 = \log\frac{1 + e^{L_1} e^{L_2}}{e^{L_1} + e^{L_2}}. \quad (10)$$

By $\tilde{\boxplus}$ we will denote the approximation of (10) introduced in [3]

$$L_1 \boxplus L_2 \approx L_1 \tilde{\boxplus} L_2 = \mathrm{sgn}(L_1)\,\mathrm{sgn}(L_2)\,\min(|L_1|, |L_2|). \quad (11)$$

Using (11) recursively, (9) can be approximated by

$$\sum_{j=1}^{J} \boxplus L_j \approx \sum_{j=1}^{J} \tilde{\boxplus} L_j = \left(\prod_{j=1}^{J} \mathrm{sgn}(L_j)\right) \cdot \min_{j=1\ldots J} |L_j|. \quad (12)$$

If (12) is used for (1) in decoding, this is called the *min-sum* algorithm.

A. Part 1, Approximation of (3)

To find a better approximation of (10) we propose rewriting (10) as

$$\begin{aligned} L_1 \boxplus L_2 &= L_1 + \log(e^{-L_1} + e^{L_2}) \\ &\quad - \log(e^{L_1} + e^{L_2}). \end{aligned} \quad (13)$$

For $\log(e^{L_1} + e^{L_2})$ it was derived in [10] that

$$\begin{aligned} \log(e^{L_1} + e^{L_2}) &= \max(L_1, L_2) + \log(1 + e^{-|L_2 - L_1|}) \\ &= \max(L_1, L_2) + f_+(|L_1 - L_2|), \end{aligned} \quad (14)$$

with $f_+(x) = \log(1 + e^{-x})$ being a correction function. f_+ can be efficiently implemented using a precomputed lookup-table containing results of f_+ for a finite number of input values. In [11] it was shown that already with a single entry in the lookup-table of f_+, which then resembles merely a correction factor instead of a correction function, very good performances can be achieved in Log-MAP decoding of Turbo codes [10], where f_+ is used once with (14).

Using an approximation, denoted by $\tilde{f}_+$, twice when inserting (14) in (13) results in

$$\begin{aligned} L_1 \tilde{\boxplus} L_2 &= L_1 + \max(-L_1, L_2) + \tilde{f}_+(|-L_1 - L_2|) \\ &\quad - \max(L_1, L_2) - \tilde{f}_+(|L_1 - L_2|) \\ &= \mathrm{sgn}(L_1) \cdot \mathrm{sgn}(L_2) \cdot \min(|L_1|, |L_2|) \\ &\quad + \tilde{f}_+(|L_1 + L_2|) - \tilde{f}_+(|L_1 - L_2|), \end{aligned} \quad (15)$$

where $\tilde{\boxplus}$ indicates the usage of $\tilde{f}_+$. Using (15), (9) can be computed recursively.

An alternative approach for approximating (13) is presented in [12]. A single, but more complex correction factor $\tilde{f}_+^c(L_1, L_2)$, with

$$\tilde{f}_+^c(u, v) = \begin{cases} -c_+ & \text{if } |u+v| > 2|u-v| \\ & \text{and } |u-v| < x_+^{\lim} \\ +c_+ & \text{if } |u-v| > 2|u+v| \\ & \text{and } |u+v| < x_+^{\lim} \\ 0 & \text{otherwise} \end{cases}, \quad (16)$$

replaces the two correction factors $\tilde{f}_+$ in (13) since

$$\tilde{f}_+(|L_1 + L_2|) - \tilde{f}_+(|L_1 - L_2|) \approx \tilde{f}_+^c(L_1, L_2). \quad (17)$$

Note that with $\tilde{f}_+^c$ the lookup-table is limited to a single entry c_+.

B. Part 2, Approximation of (4)

For writing (4) using *box-plus* operations we derive a new function of the type

$$\sum_{j=1,j\neq n}^{J}\boxplus L_j = f\left(\sum_{j=1}^{J}\boxplus L_j, L_n\right) = f\left(L_\Sigma, L_n\right). \quad (18)$$

Without loss of generality we assume that L_n was the last element in the recursive computation of L_Σ in (9) and with the abbreviation $L_{j\neq n}=\sum_{j=1,j\neq n}^{J}\boxplus L_j$ we get

$$L_\Sigma = L_{j\neq n}\boxplus L_n. \quad (19)$$

Using (10) this can be transformed to

$$\begin{aligned} L_{j\neq n} &= L_\Sigma + \log(e^{-L_\Sigma} - e^{L_n}) - \log(e^{L_\Sigma} - e^{L_n}) \\ &= \operatorname{sgn}(L_n)\cdot\operatorname{sgn}(L_\Sigma)\cdot\min(|L_n|,|L_\Sigma|) \\ &\quad + f_-(|L_n + L_\Sigma|) - f_-(|L_n - L_\Sigma|), \quad (20) \end{aligned}$$

with $f_-(x) = \log(1 - e^{-x})$ being a second, new correction function, again possibly implemented by another lookup-table function (or correction factor) $\tilde{f}_-$. Since $|L_\Sigma| \leq |L_n|$ (20) can be reduced with using $\tilde{f}_-$ for f_- to

$$\begin{aligned} \tilde{L}_{j\neq n} = \;& \operatorname{sgn}(L_n)L_\Sigma + \tilde{f}_-(|L_n + L_\Sigma|) \\ & - \tilde{f}_-(|L_n - L_\Sigma|). \quad (21) \end{aligned}$$

C. Summary

Now the matrix $\underline{E}^{(i)}$ can be computed very efficiently using the lookup-table functions $\tilde{f}_+$ and $\tilde{f}_-$ by replacing (1), resp. (2), with the two parts introduced above. First for the complete rows the $\bar{s}_p^{(i)}$ are computed with $\tilde{f}_+$ as

$$\bar{s}_p^{(i)} \approx \tilde{s}_p^{(i)} = \sum_{j\in\operatorname{supp}(\underline{b}_p)}\tilde{\boxplus} Z_{pj}^{(i)} \quad (22)$$

by recursively applying (15). Afterwards the matrix entries $E_{pn}^{(i)}$ are extracted from the $\tilde{s}_p^{(i)}$ by using (21)

$$\begin{aligned} E_{pn}^{(i)} \approx \tilde{E}_{pn}^{(i)} = \operatorname{sgn}(Z_{pn}^{(i)})\cdot\tilde{s}_p^{(i)} + \tilde{f}_-(|Z_{pn}^{(i)} + \tilde{s}_p^{(i)}|) \\ - \tilde{f}_-(|Z_{pn}^{(i)} - \tilde{s}_p^{(i)}|) \quad . \quad (23) \end{aligned}$$

Thus, with the introduced approximations only additions, subtractions, sign multiplications and min operations have to be performed in the computation of the elements of matrix $\underline{E}^{(i)}$. The complete decoding process will be denoted as *lookup-sum* algorithm.

IV. Complexity Comparison

In this section the computational complexity of *min-sum* decoding ($\tilde{\tilde{\boxplus}}$) and *belief propagation*, or *sum-product*, decoding ($\boxplus$) is compared to the *lookup-sum* algorithm ($\tilde{\boxplus}$) derived in the previous section.

The algorithms $\tilde{\tilde{\boxplus}}$ and $\boxplus$ are separated into two parts similar to equations (22) and (23), which are used for $\tilde{\boxplus}$. For $\boxplus$ we considered (3) and (4) with $\tanh(x/2)=(e^x-1)/(e^x+1)$ and $2\,\operatorname{atanh}(x)=\log((1+x)/(1-x))$. The $\tilde{\tilde{\boxplus}}$ algorithm requires the determination of the two L-values with the smallest amplitude and the product of the signs.

The necessary numbers of operations for one row p of $\underline{E}^{(i)}$ are listed in Table I assuming a weight of J for row p of $\underline{B}$. For comparison the values of the log-likelihood ratio sum-product algorithm (LLR-SPA) with correction presented in [12] are included in Table I. This algorithm computes the $E_{pn}^{(i)}$ using a lookup-table in a forward-backward algorithm with three recursions on the trellis of complete rows. In contrast the *lookup-sum* algorithm $\tilde{\boxplus}$ proposed in this paper requires only a single recursion and an extraction cycle. An efficient logic circuit implementation for a correction factor (a lookup-table with a single entry) can be found in [11].

Considering additionally the rest of the decoding process which is identical for all algorithms, i.e., (5), (6) and (7), we see that using $\tilde{\boxplus}$, no multiplications, divisions, exponential or logarithmic operations are required anymore in the whole iterative decoding process.

In Table II the in total required weighted operations (wOp) for the computation of one row with weight J are given. The operations of Table I are weighted according to the weights of the ETSI basic operators

TABLE I
REQUIRED OPERATIONS FOR A ROW OF WEIGHT J

	$\tilde{\tilde{\boxplus}}$		$\tilde{\boxplus}$		LLR-SPA	$\boxplus$	
	part 1	part 2	part 1	part 2	with correction [12]	part 1	part 2
$\operatorname{sgn}(x)\cdot\operatorname{sgn}(y)$	$J-1$	-	$J-1$	-	$2\cdot(J-1)+J$	-	-
$\operatorname{sgn}(x)\cdot y$	2	J	$J-1$	J	$2\cdot(J-1)+J$	-	-
$\min(\lvert x\rvert,\lvert y\rvert)$	$J-1$	-	$J-1$	-	$2\cdot(J-1)+J$	-	-
$\tilde{f}_\pm(\lvert x\rvert)\rightarrow$ lookup-table	-	-	$2\cdot(J-1)$	$2\cdot J$	$4\cdot(J-1)+2\cdot J$	-	-
$+$ and $-$	-	-	$4\cdot(J-1)$	$4\cdot J$	$8\cdot(J-1)+4\cdot J$	$2\cdot J$	$2\cdot J$
$*$	-	-	-	-	-	$J-1$	-
$/$	-	-	-	-	-	J	$2\cdot J$
$\exp(x)$ and $\log(x)$	-	-	-	-	-	J	J

TABLE II
REQUIRED WEIGHTED OPERATIONS FOR A ROW OF WEIGHT J

	weighted operations (wOp)
$\tilde{\tilde{\boxplus}}$ (*min-sum*)	$3 \cdot J$
$\tilde{\boxplus}$ (*lookup-sum*)	$16 \cdot J - 9$
LLR-SPA w. corr. [12]	$27 \cdot J - 18$
$\boxplus$ (BP / *sum-product*)	$(59 + w_{\exp} + w_{\log}) \cdot J - 1$

in [13]. These basic operators and their weights allow an estimation of the computational demands of a fixed-point implementation without an actual hardware-dependent realization. Most operations have a weight of $w = 1$ (assuming 16 bit accuracy). The division has a weight of $w_{\text{div}} = 18$. The weights of "exp" and "log", $w_{\exp}$ resp. $w_{\log}$, depend on the complexity of their implementation, e.g., series expansion, since "exp" and "log" are not basic operators themselves.

V. SIMULATIONS RESULTS

In this section the performance of the three considered algorithms, $\boxplus$ (BP / *sum-product*), $\tilde{\boxplus}$ (*lookup-sum*) and $\tilde{\tilde{\boxplus}}$ (*min-sum*), is compared using simulations. Since they show superior performance for medium block length [8], we used DSC codes instead of standard LDPC codes. DSC codes also have a unique parity-check matrix, not requiring any optimization. Both lookup-tables have T entries, equally distributed for the input $|x|$ between 0 and $x_{\max}$. For $|x| > x_{\max}$ we set $\tilde{f}_+(|x|) = \tilde{f}_-(|x|) = 0$. AWGN serves as channel distortion and the maximum number of iterations is 50. The simulations showed that more iterations will not result in a noticeable further increase of the performance due to a usually significantly lower number of in average required iterations.

Fig. 1 depicts the bit-error rate (BER) and frame-error rate (FER) performance of the (73,45) DSC code, which has a row weight of $J = 9$. $\tilde{\boxplus}$ with $T = 16$ shows no difference for small E_b/N_0 and a barely noticeable degradation at high E_b/N_0 with respect to $\boxplus$. $\tilde{\boxplus}$ with $T = 1$, i.e., using a single correction factor, has a very small loss of about $0.05\,\text{dB}$ on the whole E_b/N_0-range. This matches the performance degradation for the more complex LLR-SPA with correction [12]. A significant deterioration of $0.5\,\text{dB}$ is caused by $\tilde{\tilde{\boxplus}}$. Fig. 2 shows the average number of iterations executed by the different algorithms. Despite its superior BER performance, $\tilde{\boxplus}$ with $T = 1$ requires approximately the same number of iterations in average as $\tilde{\tilde{\boxplus}}$. For $\tilde{\boxplus}$ with $T = 16$ only slightly more iterations need to be executed than for $\boxplus$. In Fig. 3 the average number of iterations of Fig. 2 and the weighted operations per row in an iteration of Table II are combined to obtain the average weighted operations per row. For the weighted operation of $\boxplus$ we set $w_{\exp}$ and $w_{\log}$ to the minimum value $w_{\exp} = w_{\log} = 1$. Of course, the actual demands

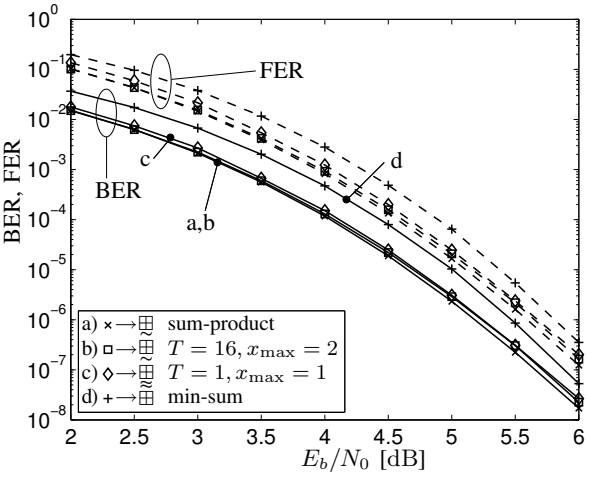

Fig. 1. BER and FER for the (73,45) DSC code

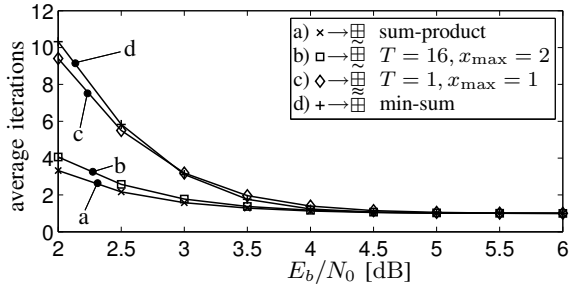

Fig. 2. Average iterations for the (73,45) DSC code

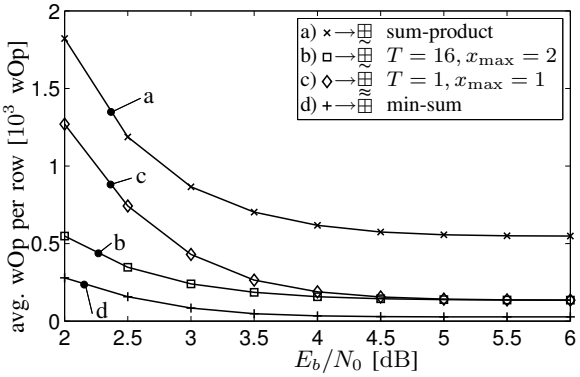

Fig. 3. Average weighted operations (wOp) per row for the (73,45) DSC code

are usually higher. A significantly lower complexity of $\tilde{\boxplus}$ compared to $\boxplus$ can be observed.

Fig. 4 depicts the BER and FER for the (273,191) DSC code. Due to the larger row weight, $J = 17$, the degradation of $\tilde{\tilde{\boxplus}}$ increases significantly to about $1\,\text{dB}$ in BER and FER, since the information of 15 out of $J - 1 = 16$ other entries in a row is omitted by the *min*-operation without correction in the computation for an element of the matrix $\underline{E}^{(i)}$. The performances of $\tilde{\boxplus}$ matches the results found for the (73,45) DSC code. $\tilde{\boxplus}$ with $T = 16$ coincides with $\boxplus$ while $\tilde{\boxplus}$ with $T = 1$ shows a small degradation below $0.1\,\text{dB}$. Fig. 5 presents the average number of required iterations for

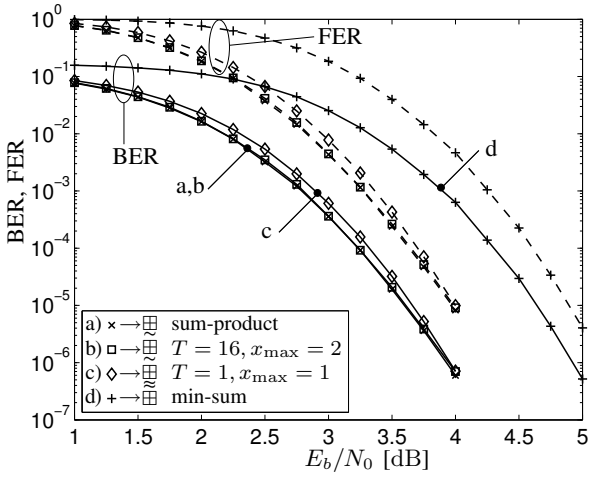

Fig. 4. BER and FER for the (273,191) DSC code

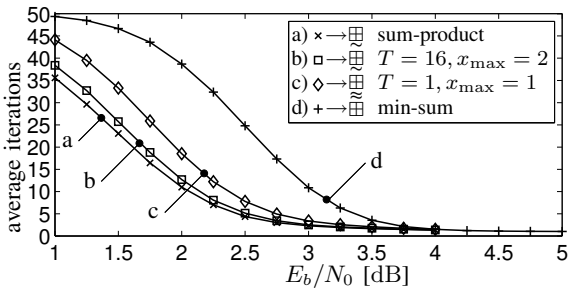

Fig. 5. Average iterations for the (273,191) DSC code

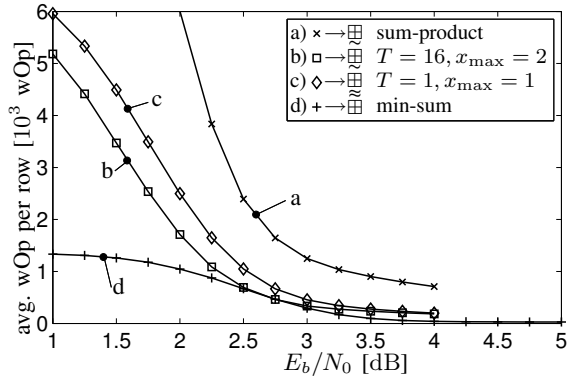

Fig. 6. Average weighted operations (wOp) per row for the (273,191) DSC code

the (273,191) DSC code. Due to the worse BER and FER performance described above, the values for $\tilde{\tilde{\boxplus}}$ are not similar $\tilde{\boxplus}$ with $T = 1$, but lie significantly above the numbers of the other shown algorithms. In Fig. 6 it can be observed that $\tilde{\boxplus}$ with $T = 16$ requires an approximately similar average number of weighted operations as $\tilde{\tilde{\boxplus}}$ in an E_b/N_0-range from 2.5 dB to 3 dB. The BER of $\tilde{\boxplus}$ in this E_b/N_0-range can be sufficient for applications such as speech coding.

VI. Conclusion

In this paper we presented a low-complexity approximation based on lookup-tables for the belief propagation algorithm. With the proposed *lookup-sum* algorithm the whole iterative process consists only of additions, subtractions, sign and min operations and table-lookups. It was shown that there is no significant difference in performance compared to standard belief propagation decoding even if the lookup-table size is very small. Already for a correction factor, i.e., a lookup-table with a single entry, the performance of the complex, non-approximated belief propagation algorithm can be approached by less than 0.1 dB in E_b/N_0. Besides the significant complexity reductions the main algorithmic contributions are:

- Introduction of a new correction function f_- and its approximation by a table-lookup or a single correction factor
- Efficient decomposition of a length J *box-plus* operation

References

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon Limit Error-Correcting Coding and Decoding," in *Proc. ICC'93 Geneva*, May 1993, pp. 1064–1070.

[2] C. Berrou and A. Glavieux, "Near Optimum Error Correcting Coding and Decoding: Turbo-Codes," *IEEE Trans. Comm.*, vol. 44, no. 10, pp. 1261–1271, Oct. 1996.

[3] J. Hagenauer, E. Offer, and L. Papke, "Iterative Decoding of Binary Convolutional Codes," *IEEE Trans. Comm.*, vol. 42, no. 2, pp. 429–445, March 1996.

[4] R. G. Gallager, "Low-Density Parity-Check Codes," *IRE Trans. Inform. Theory*, vol. 8, pp. 21–28, January 1962.

[5] D. J. C. MacKay and R. M. Neal, "Near Shannon Limit Performance of Low Density Parity Check Codes," *Electron. Lett.*, vol. 32, no. 18, pp. 1645–1646, August 1996.

[6] D. J. C. MacKay, "Good Error-Correcting Codes Based on Very Sparse Matrices," *IEEE Trans. Inform. Theory*, vol. 45, no. 2, pp. 399–431, March 1999.

[7] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA: Morgan Kaufmann, 1988.

[8] R. Lucas, M. P. Fossorier, Y. Kou, and S. Lin, "Iterative Decoding of One-Step Majority Logic Decodable Codes Based on Belief Propagation," *IEEE Trans. Comm.*, vol. 48, no. 6, pp. 931–937, June 2002.

[9] E. J. Weldon, "Difference-set cyclic codes," *Bell Syst. Tech. J.*, vol. 45, pp. 1045–1055, September 1966.

[10] P. Robertson, P. Hoeher, and E. Villebrun, "Optimal and Sub-Optimal Maximum a Posteriori Algorithms Suitable for Turbo Decoding," *Europ. Trans. Telecommun. (ETT)*, vol. 8, no. 2, pp. 119–125, March-April 1997.

[11] W. J. Gross and P. G. Gulak, "Simplified MAP algorithm suitable for implementation of turbo decoders," *Electron. Lett.*, vol. 34, no. 16, pp. 1577–1578, August 1998.

[12] E. Eleftheriou, T. Mittelholzer, and A. Dholakia, "Reduced-complexity decoding algortihm for low-density parity-check codes," *Electron. Lett.*, vol. 37, pp. 102–104, January 2001.

[13] ETSI SMG11, "AMR permanent document (AMR-9): Complexity and delay assessment," ETSI Tdoc SMG11 136/98.