Comparison of reweighted message passing algorithms for LDPC decoding

Henk Wymeersch, Federico Penna, Vladimir Savic and Jun Zhao

Original Publication:

Henk Wymeersch, Federico Penna, Vladimir Savic and Jun Zhao, Comparison of reweighted message passing algorithms for LDPC decoding, 2013, IEEE International Conference on Communications (ICC), 2013.

IEEE International Conference on Communications (ICC 2013), 9-13 June 2013, Budapest, Hungary

Postprint available at: Linköping University Electronic Press
http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-88545

Comparison of Reweighted Message Passing Algorithms for LDPC Decoding

Henk Wymeersch*, Federico Penna†, Vladimir Savic‡, and Jun Zhao*
* Chalmers University of Technology, Gothenburg, Sweden, {henkw@,zjun@student.}chalmers.se
† Fraunhofer Heinrich Hertz Institute, Berlin, Germany, federico.penna@hhi.fraunhofer.de
‡ Linköping University, Linköping, Sweden, vladimir.savic@liu.se

***Abstract*—Low density parity check (LDPC) codes can be decoded with a variety of decoding algorithms, offering a trade-off in terms of complexity, latency, and performance. We describe seven distinct LDPC decoders and provide a performance comparison for a practical regular LDPC code. Our simulations indicate that the best performance/latency trade-off is achieved by one version of the reweighted max-product decoder. When latency is not an issue, the traditional sum-product decoder yields the best performance.**

I. Introduction

Low density parity check (LDPC) codes [1] exhibit capacity-approaching performance with polynomial decoding and encoding complexity. Since the introduction of message-passing decoders [2], LDPC codes have entered many wireless standards, and are the main competitor of turbo codes.

Standard decoding algorithms over the additive white Gaussian noise (AWGN) channel are based on belief propagation, including the sum-product algorithm and the max-product algorithm (or, equivalently, the min-sum algorithm when formulated in the negative logarithmic domain). These are message passing decoders operate by exchanging messages, often in the form of log-likelihood ratios (LLRs), on the Tanner graph of the code. While message passing decoders offer a good performance/complexity trade-off, they have a number of drawbacks: (i) they offer no convergence guarantees; (ii) even when the decoder converges, there are no guarantees on the quality of the final decision; (iii) they have variable decoding latency. The first and second issue are addressed by the more recent linear programming (LP) decoder, albeit at a cost in terms of complexity [3]. A reduced-complexity tree-reweighted method [4] was compared to LP decoding in [5] for repeat-accumulate codes, and analyzed for the binary symmetric channel in [6]. A detailed treatment on LP decoding and reweighted message-passing decoders can be found in [7] and references therein. However, no comparison of the various decoders for a high-rate standardized LDPC code over the AWGN channel was performed.

In this paper, we compare seven distinct decoders: the LP decoder and six message passing decoders. For each decoder, we quantify the complexity and provide numerical performance results for a binary LDPC code used in the IEEE 802.3an 10 Gb/s Ethernet standard. Our results indicate that the choice of "best" decoder depends on the performance criterion, affordable complexity, and affordable latency.

Notation: Vectors are column vectors and will be written in bold. The binary field will be denoted by $\mathbb{B}$, $\sum_{\sim\{x_n\}} f(\mathbf{x})$ (resp. $\max_{\sim\{x_n\}} f(\mathbf{x})$) denotes summation (resp. maximization) over all possible $\mathbf{x}$ with the n-th component fixed to the value x_n, and $p(x_n|\mathbf{y})$ is shorthand for $p(X_n = x_n|\mathbf{Y} = \mathbf{y})$.

II. Problem Formulation

A. Transmission Model

We consider an error-correcting code of rate K/N, with sparse parity check matrix $\mathbf{H}$. At the transmitter, we encode a sequence of K bits, leading to a codeword $\mathbf{x} \in \mathcal{C} \subset \mathbb{B}^N$, $N > K$, where $\mathcal{C}$ is the set of all binary sequences of length N satisfying the parity check constraints

$$\mathbf{x} \in \mathcal{C} \iff \mathbf{H}\mathbf{x} = \mathbf{0}. \tag{1}$$

Assuming binary phase shift keying and transmission over an AWGN channel, we observe

$$\mathbf{y} = 2\mathbf{x} - \mathbf{1} + \mathbf{n}, \tag{2}$$

where $\mathbf{n}$ is a sequence of N i.i.d. AWGN samples with variance σ^2. The observation is transformed to a sequence of N log-likelihood ratios $\boldsymbol{\lambda}_{\text{ch}} \in \mathbb{R}^N$, with

$$\lambda_{\text{ch},n} = \log \frac{p(y_n|x_n = 1)}{p(y_n|x_n = 0)} \tag{3}$$

$$= 2\frac{y_n}{\sigma^2}. \tag{4}$$

The objective of the receiver is to recover $\mathbf{x}$ from $\mathbf{y}$, or equivalently, from $\boldsymbol{\lambda}_{\text{ch}}$. We will assume $p(\mathbf{x})$ is constant in $\mathcal{C}$, and zero elsewhere.

B. Decoding Strategies

Ideally, we aim for a decoder that minimizes the word error probability (WEP). The optimal decoder is the maximum a posteriori (MAP) decoder, given by

$$\hat{\mathbf{x}}_{\text{W}}(\mathbf{y}) = \arg\max_{\mathbf{x}\in\mathbb{B}^N} p(\mathbf{x}|\mathbf{y}) \tag{5}$$

$$= \arg\max_{\mathbf{x}\in\mathcal{C}} \prod_{n=1}^{N} p(y_n|x_n). \tag{6}$$

In practice, we are often satisfied with a decoder that minimizes the bit error probability (BEP). The optimal decoder is the bit-wise maximum a posteriori decoder, given by

$$\hat{\mathbf{x}}_{\text{B}}(\mathbf{y}) = \left[\hat{x}_{\text{B},1}(\mathbf{y}), \hat{x}_{\text{B},2}(\mathbf{y}), \ldots, \hat{x}_{\text{B},N}(\mathbf{y})\right]^T, \tag{7}$$

where

$$\hat{x}_{\text{B},n}(\mathbf{y}) = \arg\max_{x_n\in\mathbb{B}} p(x_n|\mathbf{y}). \tag{8}$$

For most practical codes (with the exception of very short codes and convolutional codes), both (5) and (8) are mathematically

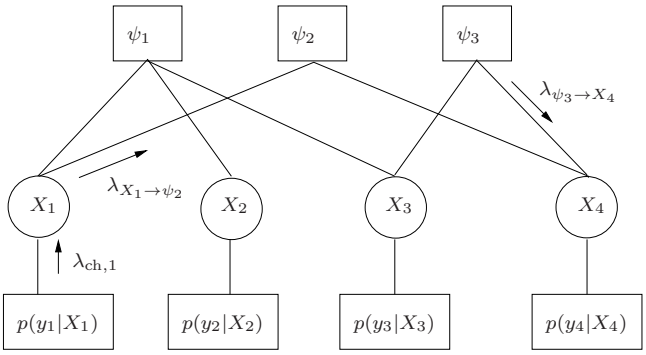

Figure 1. Factor graph corresponding to the parity check matrix $\mathbf{H} = [\mathbf{h}_1\, \mathbf{h}_2\, \mathbf{h}_3]^T$, with $\mathbf{h}_1^T = [1\,1\,1\,0]$, $\mathbf{h}_2^T = [1\,0\,0\,1]$, and $\mathbf{h}_3^T = [0\,0\,1\,1]$. Hence, check 1 involves variables X_1, X_2, and X_3, so that $\mathbf{x}_1 = [x_1\, x_2\, x_3]^T$, $\mathbf{x}_2 = [x_1\, x_4]^T$, and $\mathbf{x}_3 = [x_3\, x_4]^T$.

intractable, due to the high-dimensional optimization in (5) or the high-dimensional marginalization in (8). For this reason, one must resort to practical, sub-optimal decoders, as described in the next section.

III. Practical Decoders

Many practical decoders can be interpreted as message passing algorithm on a factor graph representation of $p(\mathbf{x}|\mathbf{y})$ [8]. This distribution can be factorized as

$$\begin{aligned} p(\mathbf{x}|\mathbf{y}) &\propto p(\mathbf{y}|\mathbf{x})p(\mathbf{x}) \\ &\propto \prod_{n=1}^{N} p(y_n|x_n) \prod_{l=1}^{N-K} \underbrace{\mathbb{I}\left\{\mathbf{h}_l^T\mathbf{x}=0\right\}}_{\doteq \psi_l(\mathbf{x}_l)}, \end{aligned} \tag{9}$$

where $\mathbf{h}_l^T$ is the l-th row of $\mathbf{H}$. Since $\mathbf{H}$ is sparse, the factor $\mathbb{I}\left\{\mathbf{h}_l^T\mathbf{x}=0\right\}$, referred to as a *check*, only depends on very few of the entries in $\mathbf{x}$, denoted by $\mathbf{x}_l$. The factor graph corresponding to (9) for a small matrix $\mathbf{H}$ is shown in Fig. 1. We denote by $\mathcal{N}(X_n)$ the indices of the checks in which the variable X_n appears. Similarly, we denote by $\mathcal{N}(\psi_l)$ the indices of the variables that appear in the l-th check. Finally, we denote by d_c the average check node degree (the average size of $\mathcal{N}(\psi_l)$) and by d_v the average variable node degree.

We now describe seven distinct decoders, and explicitly describe the decoding algorithms.

A. Sum-Product Decoder (SPD)

The sum-product algorithm is a message passing algorithm on a factor graph that aims to compute the marginals used in (8), i.e.,

$$p(x_n|\mathbf{y}) = \sum_{\sim\{x_n\}} p(\mathbf{x}|\mathbf{y}),\ \forall n. \tag{10}$$

The message from variable vertex X_n to factor vertex ψ_l is given by

$$\mu_{X_n\to\psi_l}(x_n) = p(y_n|x_n) \prod_{k\in\mathcal{N}(X_n)\setminus\{l\}} \mu_{\psi_k\to X_n}(x_n), \tag{11}$$

where the messages $\mu_{\psi_k\to X_n}(x_n)$ are initialized to be uniform distributions. The message from factor vertex ψ_l to variable vertex X_n is given by

$$\mu_{\psi_l\to X_n}(x_n) \propto \sum_{\sim\{x_n\}} \psi_l(\mathbf{x}_l) \prod_{m\in\mathcal{N}(\psi_l)\setminus\{n\}} \mu_{X_m\to\psi_l}(x_m). \tag{12}$$

At any iteration, we also have approximations of the marginals

$$b_{X_n}(x_n) \propto p(y_n|x_n) \prod_{k\in\mathcal{N}(X_n)} \mu_{\psi_k\to X_n}(x_n). \tag{13}$$

The quality of the approximation (i.e., how well $b_{X_n}(x_n)$ approximates $p(x_n|\mathbf{y})$) depends mainly on the girth of the LDPC code. For numerical stability, messages are often computed in the logarithmic domain, replacing $\mu_{A\to B}(x)$ with $\log\mu_{A\to B}(x)$. The message passing rules are found by replacing products by sums, and sums by the $\max^*$-operation[1] in (11)–(13) [9], [10]. Gains in storage can be achieved by representing the messages as LLRs $\lambda_{A\to B} = \log\mu_{A\to B}(1) - \log\mu_{A\to B}(0)$, leading to the following message passing rules. The message from variable vertex X_n to factor vertex ψ_l is given by

$$\lambda_{X_n\to\psi_l} = \lambda_{\mathrm{ch},n} + \sum_{k\in\mathcal{N}(X_n)\setminus\{l\}} \lambda_{\psi_k\to X_n}, \tag{14}$$

where $\lambda_{\psi_k\to X_n}$ is initialized to zero (the LLR equivalent of a uniform distribution). The message from factor vertex ψ_l to variable vertex X_n is given by

$$\lambda_{\psi_l\to X_n} = f_{\max^*}\left(\{\lambda_{X_m\to\psi_l}\}_{m\neq n}\right), \tag{15}$$

where, since $\log\mu_{X_m\to\psi_l}(x_m) = (-1)^{(1-x_m)}\lambda_{X_m\to\psi_l}/2$,

$$\begin{aligned} &f_{\max^*}\left(\{\lambda_{X_m\to\psi_l}\}_{m\neq n}\right) = \\ &\max^*_{\mathbf{x}_l:x_n=1}\left\{\log\psi_l(\mathbf{x}_l) + \frac{1}{2}\sum_{m\in\mathcal{N}(\psi_l)\setminus\{n\}}(-1)^{1-x_m}\lambda_{X_m\to\psi_l}\right\} \\ &-\max^*_{\mathbf{x}_l:x_n=0}\left\{\log\psi_l(\mathbf{x}_l) + \frac{1}{2}\sum_{m\in\mathcal{N}(\psi_l)\setminus\{n\}}(-1)^{1-x_m}\lambda_{X_m\to\psi_l}\right\}, \end{aligned} \tag{16}$$

and the beliefs become

$$\lambda_{\mathrm{b},n} = \lambda_{\mathrm{ch},n} + \sum_{k\in\mathcal{N}(X_n)} \lambda_{\psi_k\to X_n}. \tag{17}$$

A decision on the n-th bit is made as $\hat{x}_n = 1$ when $\lambda_{\mathrm{b},n} \geq 0$ and $\hat{x}_n = 0$ otherwise.

Complexity: The decoding complexity per iteration scales as $\mathcal{O}(Nd_\mathrm{v})$ additions for (14), and $\mathcal{O}((N-K)d_\mathrm{c})$ $\max^*$-operations for (15).[2] Each $\max^*$-operation requires a maximization, an addition, and a look-up operation.

[1] The $\max^*$-operation, sometimes referred to as the Jacobian logarithm, is given by $\max_i^* L_i \doteq \log\sum_{m=1}^{M} e^{L_m}$. If $\mathbf{z}$ is binary vector with B bits and f a generic function of $\mathbf{z}$, we use the notation $\max_{\mathbf{z}}^* f(\mathbf{z}) \doteq \max^*[f(\mathbf{z}_1), \ldots, f(\mathbf{z}_M)]$, where $\mathbf{z}_1$ through $\mathbf{z}_M$ are all possible ($M = 2^B$) values of $\mathbf{z}$. The $\max^*$-operation can be implemented recursively as $\max^*[L_1, L_2, \ldots, L_M] = \max^*[\max^*[L_1, L_2, \ldots, L_{M-1}], L_M]$, with $\max^*[L_1, L_2] = \max[L_1, L_2] + \log(1 + e^{-|L_1 - L_2|})$.

[2] At first glance, (15) requires $\mathcal{O}((N-K)2^{d_c})$ $\max^*$-operations. However, using an internal trellis representation of $\psi_l(\mathbf{x}_l)$ and running a forward-backward algorithm, this complexity can be reduced to $\mathcal{O}((N-K)d_\mathrm{c})$ [10, p. 161].

B. Max-Product Decoder (MPD)

The max-product algorithm is a message passing algorithm on a factor graph that aims to compute approximations of the max-marginals of the function $p(\mathbf{x}|\mathbf{y})$ [11]:

$$q(x_n|\mathbf{y}) = \max_{\sim\{x_n\}} p(\mathbf{x}|\mathbf{y}),\ \forall n. \tag{18}$$

Assuming that $p(\mathbf{x}|\mathbf{y})$ has a unique maximum, then $\hat{x}_n = \arg\max_{x_n} q(x_n|\mathbf{y})$ is equal to the n-th component of $\hat{\mathbf{x}}_{\mathrm{W}}(\mathbf{y})$, allowing us to solve (5). In the LLR-domain, the message passing rules are the same as (14)–(17), but with $\max^*$ replaced with $\max$, so that $\lambda_{\psi_l \to X_n} = f_{\max}\left(\{\lambda_{X_m \to \psi_l}\}_{m \neq n}\right)$.

Complexity: The decoding complexity per iteration scales as $\mathcal{O}(Nd_{\mathrm{v}})$ additions for (14), and $\mathcal{O}((N-K)d_{\mathrm{c}})$ $\max$-operations for (15). Note that a $\max$-operation is less complex than a $\max^*$-operation.

C. Linear Programming Decoder (LPD)

Due to the Gaussian nature of the noise we can express $\hat{\mathbf{x}}_{\mathrm{W}}(\mathbf{y})$ as

$$\hat{\mathbf{x}}_{\mathrm{W}}(\mathbf{y}) = \arg\min_{\mathbf{x}\in\mathcal{C}} \|\mathbf{y} - (2\mathbf{x} - \mathbf{1})\|^2 \tag{19}$$

$$= \arg\max_{\mathbf{x}\in\mathcal{C}} \mathbf{x}^T \boldsymbol{\lambda}_{\mathrm{ch}}. \tag{20}$$

This is an integer LP. By relaxing the set $\mathcal{C}$ to a set $\mathcal{P}$, where $\mathcal{P} \subset [0,1]^N$ and $\mathcal{P} \cap \{0,1\}^N = \mathcal{C}$, we obtain a standard linear program, which can be solved using a standard LP solver. Implementation details are deferred to the Appendix. The LP decoder has the property that when the solution $\mathbf{x}^* \in \mathcal{C}$, we are guaranteed to have found the MAP solution $\hat{\mathbf{x}}_{\mathrm{W}}(\mathbf{y})$.

Complexity: The optimization variable of the LP decoder is of dimensionality $\mathcal{O}(N2^{d_{\mathrm{c}}})$ with $\mathcal{O}(Nd_{\mathrm{c}})$ equality constraints and $\mathcal{O}(N2^{d_{\mathrm{c}}})$ inequality constraints (see (25)). For large d_{c}, explicit enumeration of the constraints is intractable, and suitable implicit methods can be used [12].

D. Reweighted Sum-Product Decoder (R-SPD)

From the tree-reweighted sum-product (TRW-SP) algorithm [13], an alternative decoding algorithm can be constructed that also computes the (sum-)marginals of $p(\mathbf{x}|\mathbf{y})$. The TRW-SP algorithm was originally developed for graphical models with pairwise interactions only. Hence, we convert the factor graph to a Markov random field with pairwise interactions (for more details, refer to [14]). In addition, we assign a constant reweighting factor ($\rho \in [0,1]$) to all edges in the graph, in contrast to original TRW-SP, where factors ρ's are computed as edge appearance probabilities (EAP) in the spanning tree polytope of the graph.[3] After some manipulations, we find the following message passing rules. The message from variable vertex X_n to factor vertex ψ_l is given by

$$\lambda_{X_n \to \psi_l} = \lambda_{\mathrm{ch},n} + \rho \sum_{k \in \mathcal{N}(X_n)\setminus\{l\}} \lambda_{\psi_k \to X_n} - (1-\rho)\lambda_{\psi_l \to X_n}. \tag{21}$$

[3]We refer to this approach as *uniformly reweighted (URW) BP*, [14], [15]. The choice of URW-BP is motivated primarily by complexity reasons (optimizing EAPs for every single edge would be prohibitive in practice), and also by the fact that regular LDPC codes by definition have a regular structure, hence the assumption of constant EAP is a reasonable approximation.

The message from factor vertex ψ_l to variable vertex X_n is given by

$$\lambda_{\psi_l \to X_n} = f_{\max^*}\left(\{\rho\lambda_{X_m \to \psi_l}\}_{m \neq n}\right) - (1-\rho)\lambda_{X_n \to \psi_l} \tag{22}$$

and the final beliefs are computed as

$$\lambda_{b,n} = \lambda_{\mathrm{ch},n} + \rho \sum_{k \in \mathcal{N}(X_n)} \lambda_{\psi_k \to X_n}. \tag{23}$$

Observe that when $\rho = 1$, R-SPD reverts to SPD.

Complexity: The decoding complexity per iteration is similar to SPD: (21) requires $\mathcal{O}(N)$ multiplications and $\mathcal{O}(Nd_{\mathrm{v}})$ additions; (22) requires $\mathcal{O}(N)$ multiplications and $\mathcal{O}((N-K)d_{\mathrm{c}})$ $\max^*$-operations.

E. Reweighted Max-Product Decoder (R-MPD)

In [6], another reweighted LDPC decoder was described to compute max-marginals. This decoder has the same structure of a tree-reweighted max-product (TRW-MP) algorithm [4], [16] with a constant weight ρ. The final decoder is given by (21)–(23), with $f_{\max^*}\left(\{\rho\lambda_{X_m \to \psi_l}\}_{m \neq n}\right)$ replaced by $f_{\max}\left(\{\rho\lambda_{X_m \to \psi_l}\}_{m \neq n}\right) = \rho f_{\max}\left(\{\lambda_{X_m \to \psi_l}\}_{m \neq n}\right)$. When $\rho = 1$, R-MPD reverts to MPD.

Complexity: The decoding complexity per iteration is similar to MPD: $\mathcal{O}(N)$ multiplications and $\mathcal{O}(Nd_{\mathrm{v}})$ additions for messages from variable nodes to factor nodes; $\mathcal{O}(N)$ multiplications and $\mathcal{O}((N-K)d_{\mathrm{c}})$ $\max$-operations for messages from factor nodes to variable nodes.

F. Reweighted Sum-Product Decoder – Version 2 (R-SPD-II)

In [15], we described an LDPC decoder, based on the uniformly reweighted sum-product algorithm. Contrary to R-SPD, we did not convert the factor graph to a graph with only pairwise interactions. The message passing rules turn out to be slightly different from R-SPD. The message $\lambda_{X_n \to \psi_l}$ is given by (21), and the belief $\lambda_{b,n}$ by (23), but now (22) is replaced by

$$\lambda_{\psi_l \to X_n} = f_{\max^*}\left(\{\lambda_{X_m \to \psi_l}\}_{m \neq n}\right), \tag{24}$$

which does not depend on ρ. Similar to R-SPD, R-SPD-II reverts to SPD when $\rho = 1$.

Complexity: The decoding complexity per iteration is slightly less than R-SPD: (21) requires $\mathcal{O}(N)$ multiplications and $\mathcal{O}(Nd_{\mathrm{v}})$ additions; (24) requires $\mathcal{O}((N-K)d_{\mathrm{c}})$ $\max^*$-operations.

G. Reweighted Max-Product Decoder – Version 2 (R-MPD-II)

A natural modification to R-SPD-II is to replace $\max^*$ by a $\max$. This results in a decoder with lower complexity. Similar to R-MPD, when $\rho = 1$, R-MPD-II reverts to MPD.

Complexity: The decoding complexity per iteration is slightly less than R-MPD: $\mathcal{O}(N)$ multiplications and $\mathcal{O}(Nd_{\mathrm{v}})$ additions for messages from variable nodes to factor nodes; $\mathcal{O}((N-K)d_{\mathrm{c}})$ $\max$-operations for messages from factor nodes to variable nodes.

IV. Performance Results

Among the seven decoders presented above, it is not clear which one offers the best performance for a given number of iterations. Since closed-form analytical performance results are hard to obtain, we resort to Monte Carlo simulations to estimate the BEP and WEP of all seven algorithms for a regular LDPC code from an IEEE standard.

A. Simulation Setup

We consider a sparse $\mathbf{H}$ with $K = 1664$ and $N = 2048$, with the constant variable node degree of 6 and a constant check node degree of 32, as adopted in the IEEE 802.3an standard [17]. Following [6], for this LDPC code, $\rho \approx 0.2$ is a meaningful choice. We have always transmitted the all-zero codeword. The six message passing decoders are run up to 1000 iterations, and a tentative decoding was performed at every iteration. When a codeword was found, decoding was aborted. For the LP decoder, we used a standard LP solver, so there is no concept of iterations.

B. Results for fixed SNR

We first present the BEP and WEP after 1000 decoding iterations, in Figs. 2 and 3, respectively, for a signal-to-noise-ratio (SNR) of 6 dB (results for lower SNR were similar, as were the optimal values of ρ), corresponding to $\sigma^2 = 0.2512$. Recall that the results for SPD and MPD correspond to $\rho = 1$, using any of the two associated reweighted methods. In terms of the BEP, we see that R-SPD-II offers the lowest BEP, followed by R-SPD. These two algorithms offer better performance as ρ gets closer to 1. R-SPD and R-SPD-II result in degraded performance for $\rho < 1$. Traditional MPD offers poor performance compared to traditional SPD. However, two reweighted methods R-MPD and R-MPD-II perform better compared to the traditional MPD. Especially R-MPD-II offers BEP performance improvements for all $\rho \in [0.1, 1]$. R-MPD also offers good BEP performance for the more limited range $\rho \in [0.1, 0.3]$. Note that in both cases, these intervals contain the predicted value of $\rho \approx 0.2$. The conclusions are similar when we consider the WEP, except that the gains in the interval $\rho \in [0.1, 0.3]$ are smaller. For $\rho \in [0.3, 0.6]$, the R-MPD decoder exhibits oscillating behavior resulting in poor performance. Note also that LP decoding outperforms the MPD message passing methods in terms of BEP, but not in WEP.

Secondly, we reduce the number of iterations to 20, which is a more practical value for hardware implementations [18]. Figs. 4 and 5 show the BEP and WEP, respectively. For SPD, the conclusions remain unaltered, while for SPD-II, the optimal value of ρ is reduced to around 0.95. R-MPD-II outperforms MPD for all $\rho \in (0, 1)$, due to faster convergence of the reweighted methods. R-MPD offers a slight improvement over MPD for $\rho \in [0.1, 0.3]$. Overall, among the MPD methods, MPD-II clearly has the best performance.

C. Results for varying SNR

From the results above, we can conclude that the reweighted methods are mainly beneficial for max-product decoding at a low number of iterations. To evaluate the effect of reweighted message passing as a function of the SNR for fixed values of ρ, we display the BEP as a function of SNR in Fig. 6. As

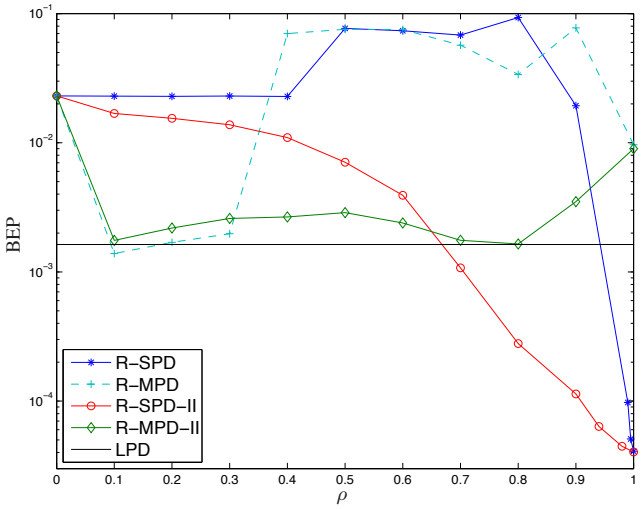

Figure 2. BEP after 1000 iterations as a function of ρ.

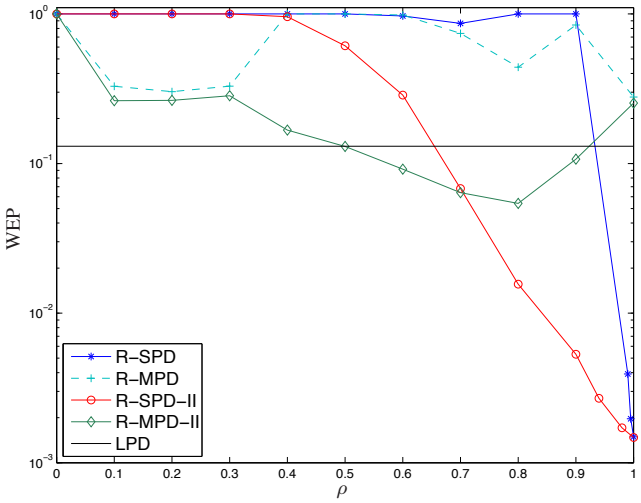

Figure 3. WEP after 1000 iterations as a function of ρ.

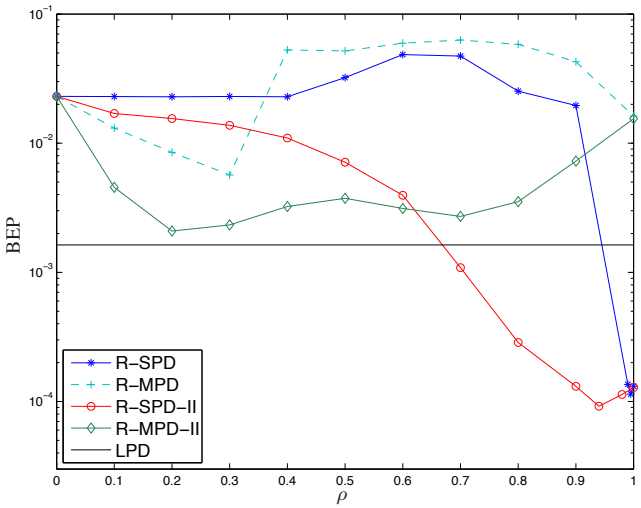

Figure 4. BEP after 20 iterations as a function of ρ.

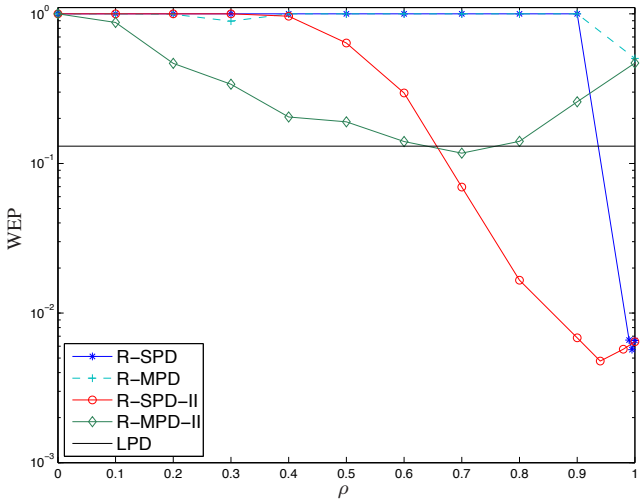

Figure 5. WEP after 20 iterations as a function of ρ.

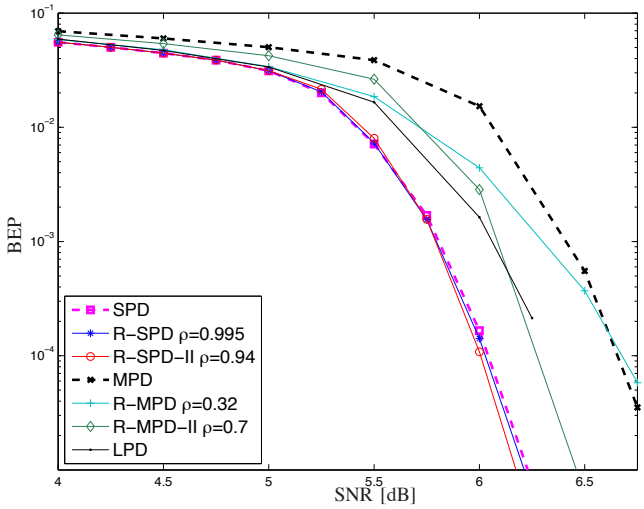

Figure 6. BEP after 20 iterations as a function of the SNR, with optimized values of ρ.

expected, we see that there is only a small gain when sum-product decoding is employed, while large gains are visible for max-product decoding. In terms of the WEP, similar conclusions hold (results not shown), and R-MPD-II achieves a performance close to LP decoding. Note that performance of the reweighted schemes can further be improved by adapting the value of ρ to the SNR.

V. Conclusions

In this paper we have compared seven LDPC decoders: the LP decoder and six message passing decoders. Four of the message passing decoders are based on tree-reweighted message passing (with free parameter $\rho \in [0, 1]$). We evaluated these decoders in terms of BEP and WEP, for up to 1000 iterations. We found that when a large number of iterations are allowed, traditional sum-product decoding offers the best performance. However, when only a small number of decoding iterations are feasible, the reweighted methods offer the best performance, for a non-trivial value of ρ, at a modest increase in complexity compared to the standard algorithms (corresponding to $\rho = 1$).

Our results and conclusions turn out to hold for a wide variety of regular LDPC codes, with shorter codes offering additional gains for the reweighted sum-product decoders [14]. However, for irregular codes more sophisticated reweighing methods should be employed [19], [20].

Acknowledgments

This work was supported, in part, by the Swedish Research Council, under grant No. 2010-5889; the European Research Council, under grant No. 258418; the Swedish Foundation for Strategic Research (SSF) and ELLIIT. Simulations were performed on resources provided by the Swedish National Infrastructure for Computing (SNIC) at C3SE.

Appendix

For short codes, our LP decoder is implemented in the primal domain following [3, (4)–(8)], while for longer codes, we resort to the alternating direction of multipliers method, described in [12, Algorithm 1]. The original problem is

$$\begin{aligned} \text{minimize} \quad & -\mathbf{x}^T \boldsymbol{\lambda}_{\text{ch}} \\ \text{s.t.} \quad & \mathbf{h}_l^T \mathbf{x} = 0, \forall l \\ & \mathbf{x} \in \{0,1\}^N. \end{aligned}$$

Note that every parity check constraint $\mathbf{h}_l^T \mathbf{x} = 0$ is an equality constraint in the binary field. This is changed to a constraint in the real field by introducing an additional *real* variable $\mathbf{w}_l \in \{0,1\}^{2^{d_c-1}}$, and two *real* matrices, $\mathbf{S}_l \in \mathbb{R}^{d_c \times N}$ and $\mathbf{C}_l \in \mathbb{R}^{d_c \times 2^{d_c-1}}$. The matrix $\mathbf{S}_l$ is such that $\mathbf{S}_l \mathbf{x} = \mathbf{x}_l$, i.e., it picks out the components in $\mathbf{x}$ involved in the l-th check. The matrix $\mathbf{C}_l$ contains as columns all the codewords of the l-th check. For example, when $\mathbf{h}_l^T \mathbf{x} = x_1 + x_{52} + x_{34}$, then

$$\mathbf{C}_l = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}.$$

Since $\mathbf{x}_l$ must be equal to one of the columns in $\mathbf{C}_l$, the constraint $\mathbf{h}_l^T \mathbf{x} = 0$ can be equivalently expressed as $\mathbf{S}_l \mathbf{x} = \mathbf{C}_l \mathbf{w}_l$, with $\mathbf{1}^T \mathbf{w}_l = 1$. Hence, the equivalent real integer LP is

$$\begin{aligned} \text{minimize} \quad & -\mathbf{x}^T \boldsymbol{\lambda}_{\text{ch}} \\ \text{s.t.} \quad & \mathbf{S}_l \mathbf{x} = \mathbf{C}_l \mathbf{w}_l, \forall l \\ & \mathbf{1}^T \mathbf{w}_l = 1, \forall l \\ & \mathbf{w}_l \in \{0,1\}^{2^{d_c-1}}, \forall l \\ & \mathbf{x} \in \{0,1\}^N. \end{aligned} \tag{25}$$

Grouping all the variables into a vector $\mathbf{z} = \begin{bmatrix} \mathbf{x}^T \mathbf{w}_1^T \dots \mathbf{w}_{N-K}^T \end{bmatrix}^T$ of length $L = N + (N-K)2^{d_c-1}$, aggregating all the linear equality constraints into one constraint $\mathbf{A}\mathbf{z} = \mathbf{b}$, where $\mathbf{A} \in \mathbb{R}^{d_c(N-K+1) \times L}$ and $\mathbf{b} \in \mathbb{R}^{d_c(N-K+1)}$. We relax $\mathbf{w}_l$ and $\mathbf{x}$ to be bounded between zero and one. The decoder is finally run in MATLAB using the command `linprog(r,[],[],A,b,lb,ub);` with $\mathtt{r} = \begin{bmatrix} -\boldsymbol{\lambda}_{\text{ch}}^T \mathbf{0}_{(N-K)2^{d_c-1}}^T \end{bmatrix}^T$, $\mathtt{lb} = \mathbf{0}_L$ and $\mathtt{ub} = \mathbf{1}_L$.

REFERENCES

[1] R.G. Gallager, "Low density parity-check codes," *IRE Trans. Inform. Theory*, vol. IT-8, pp. 21–29, Jan. 1962.

[2] N. Wiberg, *Codes and decoding on general graphs*. PhD thesis, Linköping University, Sweden, 1996.

[3] J. Feldman, M. Wainwright, and D. Karger, "Using linear programming to decode binary linear codes," *IEEE Trans. Inform. Theory*, vol. 51, no. 3, pp. 954–972, 2005.

[4] M. Wainwright, T. Jaakkola, and A. Willsky, "MAP estimation via agreement on trees: message-passing and linear programming," *IEEE Trans. Inform. Theory*, vol. 51, no. 11, pp. 3697–3717, 2005.

[5] J. Feldman, D. Karger, and M. Wainwright, "Linear programming-based decoding of turbo-like codes and its relation to iterative approaches," in *Proc. Allerton Conference*, vol. 40, pp. 467–477, Oct. 2002.

[6] Y. Jian and H. Pfister, "Convergence of weighted min-sum decoding via dynamic programming on trees," *preprint arXiv:1107.3177*.

[7] G. Even and N. Halabi, "On decoding irregular tanner codes with local optimality guarantees," *preprint arXiv:1107.2677*.

[8] F. Kschischang, B. Frey and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inform. Theory*, vol. 47, pp. 498–519, Feb. 2001.

[9] 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 International Conference on Communications*, vol. 2, pp. 1009–1013, June 1995.

[10] H. Wymeersch, *Iterative Receiver Design*. Cambridge University Press, 2007.

[11] S. Aji and R. McEliece, "The generalized distributive law," *IEEE Transactions on Information Theory*, vol. 46, pp. 325–353, Mar. 2000.

[12] S. Barman, X. Liu, S. C. Draper, and B. Recht, "Decomposition methods for large scale LP decoding," *preprint arXiv:1204.0556*.

[13] M. Wainwright, T. Jaakkola, and A. Willsky, "A new class of upper bounds on the log partition function," *IEEE Trans. Inform. Theory*, vol. 51, no. 7, pp. 2313–2335, 2005.

[14] H. Wymeersch, F. Penna, and V. Savic, "Uniformly reweighted belief propagation for estimation and detection in wireless networks," *IEEE Transactions on Wireless Communications*, vol. 11, pp. 1587–1595, April 2012.

[15] H. Wymeersch, F. Penna, and V. Savic, "Uniformly reweighted belief propagation: A factor graph approach," in *IEEE International Symposium on Information Theory*, Aug. 2011.

[16] B. Frey and R. Koetter, *Exact inference using the attenuated max-product algorithm*. Advanced mean field methods: Theory and Practice, MIT Press, 2000.

[17] Online at http://standards.ieee.org/about/get/802/802.3.html.

[18] G. Falcao, V. Silva, L. Sousa, and J. Marinho, "High coded data rate and multicodeword WiMAX LDPC decoding on Cell/BE," *Electronics Letters*, vol. 44, no. 24, pp. 1415–1416, 2008.

[19] J. Liu and R. de Lamare, "Low-latency reweighted belief propagation decoding for LDPC codes," *IEEE Communications Letters*, vol. 16, no. 10, pp. 1660–1663, 2012.

[20] J. Liu, R. de Lamare, and H. Wymeersch, "Locally-optimized reweighted belief propagation for decoding finite-length LDPC codes," in *Proc. IEEE Wireless Communications and Networking Conference (WCNC)*, Apr. 2013.