A Generalization of Residual Belief Propagation for Flexible Reduced Complexity LDPC Decoding

Moritz Beermann, Peter Vary
Inst. of Comm. Systems and Data Processing (ind)
RWTH Aachen University, Germany
{beermann|vary}@ind.rwth-aachen.de

***Abstract*—Beside the well known iterative Belief Propagation algorithm several alternative decoding schemes for Low-Density Parity-Check (LDPC) codes providing better performance in terms of residual error rate, convergence speed or computational complexity have been developed in the last years. Recently, *Informed Dynamic Scheduling* has been proposed in [1] providing different decoding strategies that dynamically decide which messages are passed throughout the decoding process. It was shown that the overall convergence can be sped up considerably and also more errors can be corrected compared to other (non-dynamic) decoding strategies. However, these improvements are somehow overshadowed by a significant amount of additional computational complexity that is needed for the selection of the messages to be updated in each decoding step. We propose two novel dynamic decoding strategies that allow for a flexible adaptation of the decoder's dynamics and reduce the additional complexity remarkably while maintaining, and in some cases even exceeding, the convergence speed and error rate performance of currently known dynamic schedules.**

I. Introduction

Low-Density Parity-Check (LDPC) codes, as originally presented in [2] and rediscovered in [3], are usually decoded using the iterative *Belief Propagation* (BP) decoding algorithm [4]. Standard BP decoding uses the so called *flooding* schedule to pass messages between variable nodes and check nodes of the code's factor graph. Due to the cycles in this graph the performance of BP decoding is suboptimal and convergence to the maximum likelihood solution can not be guaranteed. In the last years manifold variants of the BP algorithm have been developed aiming for better error rate performance, faster convergence, lower decoding complexity or combinations thereof. Prominent examples include the min-sum approximation [5], probabilistic techniques [6], and several sequential message passing schedules, e.g., [7], [8], [9].

Recently, Casado et al. presented what they termed *Informed Dynamic Scheduling* (IDS) [10], [1] as the first message passing schedule for LDPC codes that dynamically updates those parts of the graph that have not converged yet. IDS determines the order of messages to be updated and propagated using a metric called *residual* measuring how far a message is from convergence. It was shown that this dynamic approach has the capability of converging with much fewer message updates than other schedules and, in addition, is able to solve many trapping set errors [11].

The shortcoming of IDS, however, is that the computation of residuals and the selection of the messages to be updated cause a significant increase in computational complexity. Much more residuals are calculated than messages are updated and for each message update all residuals have to be searched through beforehand. While there are approaches that reduce the overhead caused by the computation of residuals, to the best of our knowledge, no efforts have yet been made to reduce the complexity of the continuous search for large residuals.

We propose two novel decoding algorithms that address the issue of computational complexity by significantly reducing the number of search operations needed per iteration. Furthermore, we show that the proposed algorithms maintain and in some cases even exceed the convergence speed and also the error rate performance of currently known IDS strategies.

II. Belief Propagation Decoding

To establish a basis of notation, we will shortly review Belief Propagation (BP) decoding. The (N, K) LDPC code (length N, K information symbols) is characterized by a parity check matrix $\mathbf{H}$ of dimension $M \times N$ (with $M = N - K$). H_{mn} denotes the entry of $\mathbf{H}$ at row m and column n. The set $\mathcal{N}(m) = \{n : H_{mn} \neq 0\}$ denotes the symbols that participate in parity check equation m, $m = 1, \ldots, M$. Similarly, the set $\mathcal{M}(n) = \{m : H_{mn} \neq 0\}$ contains the check equations in which symbol n participates. Exclusion is denoted by the operator "$\backslash$", e.g., $\mathcal{M}(n)\backslash m$ describes the set $\mathcal{M}(n)$ with check equation m excluded.

Let Z_n denote the received channel-related L-value for variable node n. Before starting the iterative processing, we set $v_{mn} = Z_n$ (v_{mn} denotes the information at the edge connecting variable node n with check node m). The horizontal step computes the check node related information

$$c_{mn} = 2\tanh^{-1}\left(\prod_{n' \in \mathcal{N}(m)\backslash n} \tanh\left(\frac{v_{mn'}}{2}\right)\right) \tag{1}$$

with c_{mn} denoting the information at the edge connecting check node m with variable node n. The vertical step updates the variable node information according to

$$v_{mn} = Z_n + \sum_{m' \in \mathcal{M}(n)\backslash m} c_{m'n}\,. \tag{2}$$

Additionally, for each variable node n, the hard decision $\hat{y}_n = \operatorname{sign}\{v_{mn} + c_{mn}\}$ (for an arbitrary m) is computed. Using those values the parity check equations are evaluated to determine whether decoding can be stopped.

This work was supported by the research cluster UMIC located at RWTH Aachen University, Germany, funded by the German research council DFG.

III. Informed Dynamic Scheduling

The first instance of an IDS strategy for general message passing algorithms was presented in [12] under the name of *Residual Belief Propagation* (RBP). Casado et al. adopted and extended this approach for the decoding of LDPC codes in [10], [13], [1]. Those approaches are based on the so called residual which is defined as the norm of the difference between a message before and after an update. The residual $r\left(c_{ij}\right)$ of a check-to-variable message c_{ij} is defined as [1]

$$r\left(c_{ij}\right) = \left\| c_{ij}^{\text{updated}} - c_{ij}^{\text{current}} \right\|. \tag{3}$$

The current value of the message is denoted by c_{ij}^{current} and the value if it was updated at this stage is denoted by c_{ij}^{updated}. In case of binary codes the messages are scalar L-values and the norm corresponds to the absolute value. From this definition it directly follows that messages with large residuals are still far from convergence while small residuals indicate messages that have already converged to a final state.

By only updating messages with large residuals convergence of the decoding process can be achieved with much fewer overall message computations than for traditional (flooding or serial) message passing schedules. It was also shown that certain IDS strategies can correct more errors than other schedules since they are able to solve trapping sets [11].

As mentioned above the drawback of these approaches is that they implicate a significant increase in computational complexity. In the following this will be described in more detail for the IDS strategies RBP and Node-Wise RBP (NWRBP) as proposed in [10].

A. Residual Belief Propagation (RBP)

RBP describes a greedy message passing schedule that always updates the check-to-variable-message with the largest residual. The algorithm is given in Alg. 1.

Algorithm 1 Residual Belief Propagation

```
1:  Initialize all $v_{ij} = Z_j$
2:  Initialize all $c_{ij} = 0$
3:  Compute all $r\left(c_{ij}\right)$
4:  **while** stopping rule not satisfied **do**
5:      Find message $c_{mn}$ with largest residual $r\left(c_{mn}\right)$
6:      Update message $c_{mn}$
7:      Set $r\left(c_{mn}\right) = 0$
8:      **for** $m' \in \mathcal{M}(n) \setminus m$ **do**
9:          Update message $v_{m'n}$
10:         **for** $n' \in \mathcal{N}(m') \setminus n$ **do**
11:             Compute $r\left(c_{m'n'}\right)$
12:         **end for**
13:     **end for**
14: **end while**
```

In accordance with [10], we define E executions of the main loop (lines 4 – 14) as one equivalent iteration, where E denotes the number of edges in the factor graph. With this definition in one equivalent iteration of RBP the same number of check messages is updated as in one iteration of a conventional (flooding or serial) schedule. In the following this criterion of equal numbers of check message updates is used to define an equivalent iteration for all considered schedules.

It can be seen that the additional complexity introduced by RBP is caused by the search for the largest residual (line 5) and the calculation of residuals (lines 10 – 12). Since each calculation of a residual requires the evaluation of (1) but is only needed to determine the order of message updates Approximate RBP (ARBP) was introduced in [10] to reduce the computational overhead. In ARBP the min-sum approximation [5] is used for the calculation of residuals while the actual message updates still use the exact BP equations (1) and (2). This approach was shown to perform nearly as well as the ones using exact residuals.

However, it turns out that for medium to large block lengths the complexity of the search for the lagest residual dominates the whole decoding algorithm as described in Alg. 1 since it raises the overall decoding complexity from $\mathcal{O}(N)$ to $\mathcal{O}(N^2)$. This is due to the fact that for each message update a list of length $E = \bar{d}_v \cdot N$ has to be searched through ($\bar{d}_v$ denotes the average variable node degree). In fact, if the residuals are organized as an ordered list and reordered whenever a residual changes, it is possible to implement RBP with complexity $\mathcal{O}(N \cdot \log(N))$. Nevertheless, a significant amount of the computational overhead is caused by this managing of an ordered list, especially as the block size increases.

B. Node-Wise RBP (NWRBP)

Another drawback of RBP originates from its greediness. As pointed out in [13] certain types of errors cannot easily be corrected by RBP which causes a distinct error floor behavior. A partial remedy to this behavior is accomplished by what was termed node-wise RBP (NWRBP) in [10] and node-wise scheduling (NS) in [1]. In this less greedy approach not only the message having the largest residual is updated in each step but all outgoing messages of the corresponding check node instead. For a more detailed description of NWRBP the reader is referred to [10]. It was shown that NWRBP can reach lower error rates than other strategies while converging only slightly slower than RBP.

Nevertheless, the computational overhead of calculating residuals and searching (or sorting) a list is similar to RBP. While the same number of residuals has to be calculated per iteration the number of search operations is slightly reduced since for each largest residual that was found on average $\bar{d}_c$ check messages are updated. The average check node degree $\bar{d}_c$, however, is only a small constant compared to the number of edges in the factor graph so that the overall complexity is only slightly reduced.

IV. Novel Decoder Scheduling

We propose two novel flexible approaches of IDS that are capable of significantly reducing the average number of search operations executed per iteration. At the same time both new approaches can overcome the problem of RBP's greediness to an even greater extent than NWRBP. Furthermore, while NWRBP converges slower than RBP our approaches converge as fast as RBP and show superior performance even when the maximum number of decoding iterations is very small.

The proposed schedules are based on the idea of not only updating one single message that has the largest residual $r_{\max}$

but updating all messages whose residuals are greater than an adaptive threshold $r_{\text{thresh}} = \alpha \cdot r_{\max}$ that is determined by a factor α (with $0 \leq \alpha \leq 1$) and the current value $r_{\max}$. For this purpose we introduce an updating queue $\mathcal{Q}$ that contains only messages having residuals greater than the current threshold. In the following the two novel approaches termed Lazy Queue Residual Decoding (LQRD) and Queue Residual Decoding (QRD) will be described in detail.

A. Lazy Queue Residual Decoding (LQRD)

LQRD is based on RBP and likewise updates a single check message in each execution of the main loop (lines 4 – 18 in Alg. 2). In contrast to RBP this message is not selected by searching through all residuals but simply obtained from the head of a message queue $\mathcal{Q}$. At the start of decoding $\mathcal{Q}$ is initialized and only modified if it runs empty. Every time this happens the value $r_{\max}$ of the largest residual is determined and all messages with residuals $r(c_{ij}) > \alpha \cdot r_{\max}$ are appended to $\mathcal{Q}$. Then message updates are performed until either the stopping rule is satisfied or the queue runs empty again. This procedure can be motivated as follows.

If at some stage of decoding there are large outliers in the values of the residuals LQRD will focus on updating the corresponding messages first since the threshold $r_{\text{thresh}} = \alpha \cdot r_{\max}$ for a message to be added to $\mathcal{Q}$ will be very high. On the other hand if there are no outliers RBP will spend a lot of effort on finding the largest residual which will, in fact, not be much different from e.g. the second or third largest residuals. LQRD will behave differently since the threshold r_{thresh} will be relatively small and many messages will be added to $\mathcal{Q}$. Thus, while still constantly focusing on parts of the graph that have not yet converged LQRD is far less greedy (depending, of course, on the choice of α) than RBP or even NWRBP. An algorithmic description of LQRD is given below in Alg. 2.

It can easily be seen that the factor α directly influences the frequency of $\mathcal{Q}$ running empty and consequently the amount of computational complexity that is saved for search operations. If a large value is chosen for α fewer messages are added to $\mathcal{Q}$ and accordingly it will run empty more often. At the same time, however, for the choice of a small α many messages with relatively small residuals will be updated which conflicts with the main idea of Informed Dynamic Scheduling.

LQRD can be seen as a generalization of RBP since it includes RBP as a special case for $\alpha = 1$. In this case whenever $r_{\max}$ is determined only the message with the largest residual is appended to $\mathcal{Q}$ since there can be no message with $r(c_{ij}) > \alpha \cdot r_{\max}$. After this message is updated the queue will be empty and again the largest residual is determined. For $\alpha = 0$ on the other hand every time $r_{\max}$ is determined all messages with residuals greater than zero are added to $\mathcal{Q}$ (note that messages in $\mathcal{Q}$ are in no specific order, though). In this case LQRD is similar to (non dynamic) message passing schedules that perform serial check node updates as presented, e.g., in [8], [9]. The main difference is that in each iteration the message that changes most is updated first and messages that do not change are not updated at all.

At the same time, if α changes the convergence speed and the error rate performance of LQRD change. In general a higher value of α results in faster convergence. However, as α increases the algorithm becomes more and more greedy and a similar error floor behavior can be observed as for RBP.

Algorithm 2 Lazy Queue Residual Decoding

```
 1: Initialize all $v_{ij} = Z_j$
 2: Initialize all $c_{ij} = 0$
 3: Compute all $r(c_{ij})$
 4: while stopping rule not satisfied do
 5:    if $\mathcal{Q}$ is empty then
 6:       Find message with largest residual
          $r_{\max} = \max_{\forall i,j} r(c_{ij})$ and append it to $\mathcal{Q}$
 7:       Append all messages $c_{ij}$ with $r(c_{ij}) > \alpha \cdot r_{\max}$ to $\mathcal{Q}$
 8:    end if
 9:    Remove message $c_{mn}$ from head of $\mathcal{Q}$
10:    Update message $c_{mn}$
11:    Set $r(c_{mn}) = 0$
12:    for $m' \in \mathcal{M}(n) \setminus m$ do
13:       Update message $v_{m'n}$
14:       for $n' \in \mathcal{N}(m') \setminus n$ do
15:          Compute $r(c_{m'n'})$
16:       end for
17:    end for
18: end while
```

B. Queue Residual Decoding (QRD)

The second proposed algorithm QRD is a modification of LQRD and has the potential of reducing the computational overhead even further while providing similar results in terms of convergence speed and error rate performance.

The difference between QRD and LQRD is that in the latter one the message queue is generated once at the beginning of decoding and then only regenerated if it runs empty regardless of the values of the residuals that have been computed in the meantime (thus the name ”lazy”). QRD, in contrast, allows for a message to be appended to $\mathcal{Q}$ as soon as its residual exceeds the threshold $r_{\text{thresh}} = \alpha \cdot r_{\max}$. This can be realized by simply checking whether a residual is greater than the current threshold whenever it is recomputed and if so adding the corresponding message to $\mathcal{Q}$. Additionally, in QRD a message that has already been added to $\mathcal{Q}$ is not updated if its residual has dropped below the current threshold r_{thresh} since the time it has been appended. This more instantaneous handling of the message queue lets it run empty less frequently which means that on average fewer search operations have to be executed per iteration. The exact algorithm is given in Alg. 3.

V. Simulation Results

In order to evaluate the performance of both proposed approaches, we consider the IEEE 802.16e WiMAX (576,288) rate-1/2 LDPC code. A frame of $K = 288$ equiprobable bits is encoded. The resulting bits are mapped to BPSK symbols and then transmitted. On the channel, the (modulation) symbols (with symbol energy $E_{\text{s}} = 1$) are subject to additive white Gaussian noise (AWGN) with (known) power spectral density $\sigma_n^2 = N_0/2$. The presented simulation results include six different decoding strategies: conventional BP with flooding schedule, shuffled BP as presented in [7], RBP, NWRBP, LQRD and QRD.

Algorithm 3 Queue Residual Decoding

```
 1: Initialize all $v_{ij} = Z_j$
 2: Initialize all $c_{ij} = 0$
 3: Compute all $r(c_{ij})$
 4: while stopping rule not satisfied do
 5:   if $\mathcal{Q}$ is empty then
 6:     Find message with largest residual $r_{\max} = \max_{\forall i,j} r(c_{ij})$ and append it to $\mathcal{Q}$
 7:     Append all messages $c_{ij}$ with $r(c_{ij}) > \alpha \cdot r_{\max}$ to $\mathcal{Q}$
 8:   end if
 9:   Remove message $c_{mn}$ from head of $\mathcal{Q}$
10:   if $r(c_{mn}) > \alpha \cdot r_{\max}$ then
11:     Update message $c_{mn}$
12:     Set $r(c_{mn}) = 0$
13:     for $m' \in \mathcal{M}(n) \setminus m$ do
14:       Update message $v_{m'n}$
15:       for $n' \in \mathcal{N}(m') \setminus n$ do
16:         Compute $r(c_{m'n'})$
17:         if $c_{m'n'}$ not in $\mathcal{Q}$ and $r(c_{m'n'}) > \alpha \cdot r_{\max}$ then
18:           Append message $c_{m'n'}$ to $\mathcal{Q}$
19:         end if
20:       end for
21:     end for
22:   end if
23: end while
```

A. Complexity

As described in Sec. IV LQRD and QRD are capable of reducing the additional complexity of IDS caused by the continuous search for the largest residual. In order to compare LQRD and QRD to RBP and NWRBP we counted the average number of iterations and the average number of search operations performed per iterations for the different schedules and for different values of E_b/N_0.

In the upper part of Fig. 1 the average number of times the largest residual has to be determined per iteration is depicted as a function of E_b/N_0 (note that a logarithmic scale is used for the y-axis). As explained in Sec. III-B the savings of NWRBP compared to RBP are in the order of the average check node degree $\bar{d}_c \approx 6.3$ of the employed WiMAX LDPC code. For LQRD and QRD curves are shown for values of $\alpha \in \{0.0, 0.5, 0.9, 1.0\}$ (note that the curve for LQRD and $\alpha = 1.0$ corresponds to that of RBP). The setups LQRD with $\alpha = 0.5$ and QRD with $\alpha = 0.9$ are highlighted by markers since they show the best BER performance (see Fig. 2) and, thus, will be used in the results below. The number of search operations decreases significantly as α decreases. At $E_b/N_0 = 1\,\text{dB}$, e.g., for QRD with $\alpha = 0.1$ the number of search operations is reduced by a factor of up to 12000 compared to RBP and up to 2100 compared to NWRBP.

The lower part of Fig. 1 shows the average number of equivalent iterations as defined in section III-A at varying values of E_b/N_0. For LQRD, QRD and RBP fewest iterations have to be conducted over the whole range of channel qualities. This reduced number of iterations can be seen as partial remedy to the increased complexity per iteration of the IDS strategies as compared to non dynamic schedules.

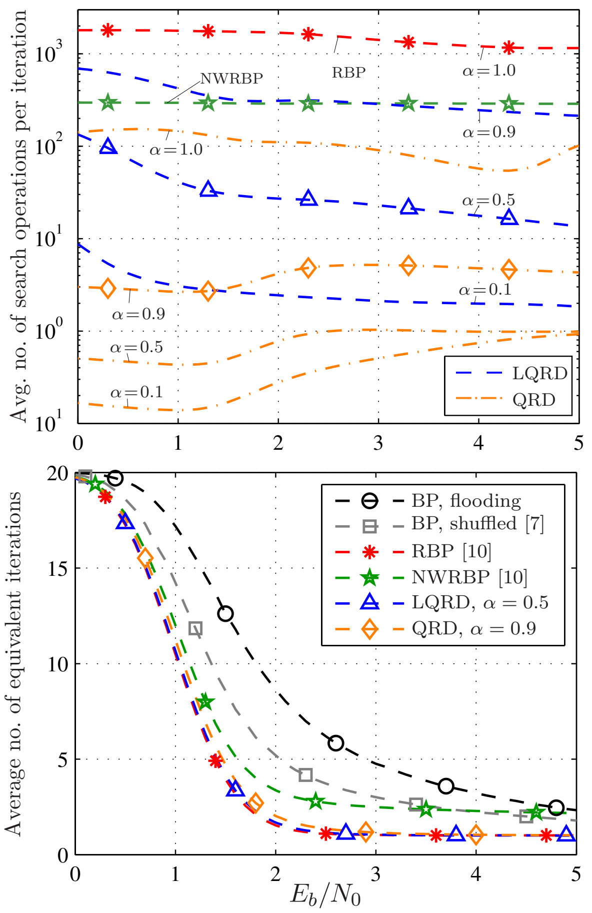

Fig. 1. Average number of equivalent iterations (lower plot) and average number of performed search operations per iteration (upper plot) needed for convergence over E_b/N_0 for WiMAX rate-1/2 blocklength 576 LDPC code

In Fig. 2 the BER performance of LQRD and QRD for values of α between 0.0 and 1.0 in steps of 0.1 is shown. While a significant complexity reduction was shown to be possible by adjusting α, the BER performance is only influenced marginally in case of QRD and slightly more in the error floor region in case of LQRD. This allows for a flexible choice of α while almost maintaining the BER performance.

All things considered LQRD and QRD are ultimately superior to the currently known IDS strategies RBP and NWRBP since they show equal and in some cases even exceeding performance while considerably reducing the additional complexity introduced by IDS.

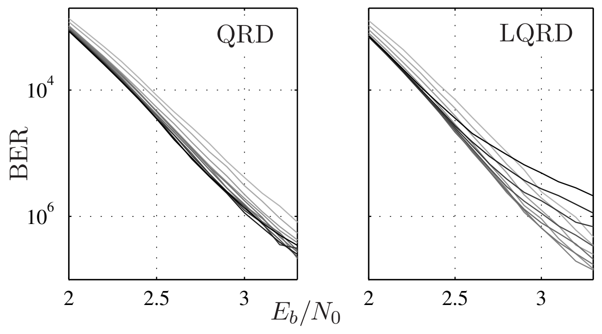

Fig. 2. Bit error rate (BER) performance of QRD (left) and LQRD (right) for values of α between 0.1 (light) and 1.0 (dark) in steps of 0.1

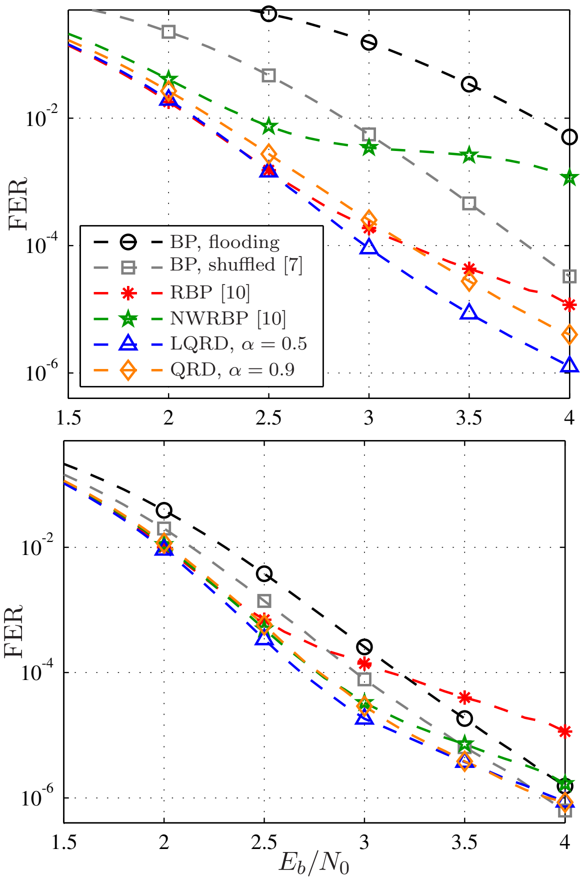

Fig. 3. FER over E_b/N_0 of the WiMAX rate-1/2 blocklength 576 LDPC code after 5 (upper) and 20 (lower) equivalent iterations

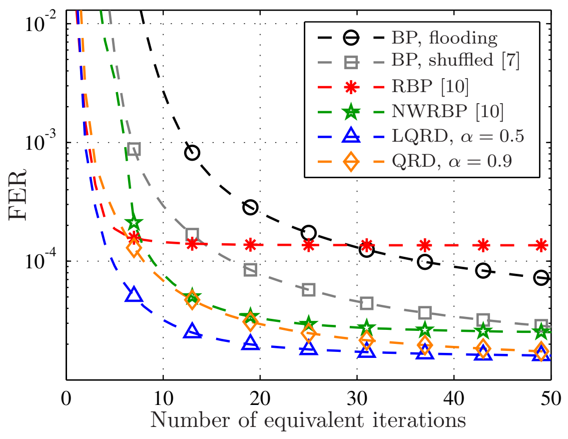

Fig. 4. FER over number of equivalent iterations at $E_b/N_0 = 3\,\text{dB}$ for the WiMAX rate-1/2 blocklength 576 LDPC code

B. Frame Error Rates and Convergence Speed

Figure 3 shows the frame error rate (FER) as a function of E_b/N_0 ($E_b = 2E_s$) after 5 (upper plot) and 20 (lower plot) equivalent iterations. It can be seen that while NWRBP performs poorly for 5 iterations RBP performs well but, in contrast to NWRBP, does hardly improve for a higher number of iterations. The performance of LQRD and QRD, however, is superior for both 5 and 20 iterations and outperforms all other decoding strategies for all shown channel qualities. It can also be seen that for the chosen values of α LQRD slightly outperforms QRD especially for smaller numbers of iterations.

Figure 4 depicts the FER over the number of equivalent iterations for $E_b/N_0 = 3\,\text{dB}$ and confirms the previous results. The convergence of QRD is almost as fast as that of RBP for smaller numbers of iterations while the performance roughly converges to that of NWRBP for higher numbers of iterations. LQRD on the other hands shows the fastest convergence and the best FER performance of all shown algorithms and for all numbers of iterations.

VI. Conclusion

We have presented two novel decoding strategies for LDPC decoding based on IDS as introduced in [10]. Both approaches address the problem of the computational overhead introduced by IDS. While the additional complexity spent on the computation of residuals can be reduced by using simplified equations to calculate approximate residuals (as, e.g., in ARBP [10]), the presented algorithms LQRD and QRD are the first approaches that significantly reduce the complexity spent on search operations, which otherwise dominates the complexity of the whole decoding process if medium to large block lengths are used.

We have shown that by not only updating the message with the largest residual in each step but instead managing a message queue that contains all messages with residuals greater than an adaptive threshold the average number of search operations can be reduced by a factor of up to 12000 for the employed WiMAX code. At the same time the convergence speed and error rate performance were shown to be equal or in some cases even better than for currently known IDS schemes.

Furthermore, the presented algorithms can be controlled by a factor α that allows for a flexible adaptation of the decoder's dynamic and influences the complexity savings as well as the convergence speed and error rate performance.

References

[1] A. Casado, M. Griot, and R. Wesel, "LDPC Decoders with Informed Dynamic Scheduling," *IEEE Trans. Comm.*, vol. 58, no. 12, Dec. 2010.

[2] R. G. Gallager, "Low-Density Parity-Check Codes," *IRE Trans. Inform. Theory*, vol. 8, no. 1, pp. 21–28, Jan. 1962.

[3] D. J. C. MacKay, "Good Error-Correcting Codes Based on Very Sparse Matrices," *IEEE Trans. Inform. Theory*, vol. 45, no. 2, Mar. 1999.

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

[5] M. Fossorier, M. Mihaljevic, and H. Imai, "Reduced Complexity Iterative Decoding of Low-Density Parity Check Codes Based on Belief Propagation," *IEEE Trans. Comm.*, vol. 47, no. 5, pp. 673–680, 1999.

[6] Y. Mao and A. Banihashemi, "A New Schedule for Decoding Low-Density Parity-Check Codes," *IEEE Global Telecommunications Conference (GLOBECOM)*, vol. 2, 2001, pp. 1007–1010.

[7] J. Zhang and M. Fossorier, "Shuffled Belief Propagation Decoding," *Proc. Ann. Asilomar Conf.*, vol. 1, Nov. 2002, pp. 8–15.

[8] M. M. Mansour and N. R. Shanbhag, "High Throughput LDPC Decoders," *IEEE Trans. VLSI Syst.*, vol. 11, no. 6, pp. 976–996, Dec. 2003.

[9] E. Sharon, S. Litsyn, and J. Goldberger, "An Efficient Message-Passing Schedule for LDPC Decoding," *Proceedings 23rd IEEE Convention of Electrical and Electronics Engineers in Israel*, Sept. 2004.

[10] A. I. V. Casado, M. Griot, and R. Wesel, "Informed Dynamic Scheduling for Belief-Propagation Decoding of LDPC Codes," *Proc. IEEE International Conference on Communications*, Glasgow, Scotland, June 2007.

[11] T. Richardson, "Error Floors of LDPC Codes," *Proc. 41st Allerton Conf. on Communication, Control, and Computing*, USA, Oct. 2003.

[12] G. Elidan, I. McGraw, and D. Koller, "Residual Belief Propagation: Informed Scheduling for Asynchronous Message Passing," *Proceedings Conference on Uncertainty in Artificial Intelligence*, USA, July 2006.

[13] A. I. V. Casado, M. Griot, and R. Wesel, "Improving LDPC Decoders via Informed Dynamic Scheduling," *Proceedings IEEE Information Theory Workshop (ITW)*, Lake Tahoe, CA, USA, Sept. 2007.