Information Forwarding in LDPC Decoding for Markov Sources

Nazia Sarwat Islam and Werner Henkel
Transmission Systems Group (TrSys)
Jacobs University Bremen
Campus Ring 1, Bremen, D - 28759.
Email: n.islam@jacobs-university.de, werner.henkel@ieee.org

***Abstract*—Belief propagation decoding of binary LDPC codes combines three independent probability estimates, a-priori, intrinsic, and extrinsic information, iterations along the Tanner graph are estimating the value of the received bits, the likelihood of the values increasing with each iteration. The a-priori estimate of the information bits is a measure of the source statistics for generating bit values 0 and 1 (mapped to ± 1), reflecting bias and redundancy in the information sequence itself. In this paper, we have modified the a-priori estimate to incorporate memory properties present in the transmitted information sequence resulting from a Markov source. We consider two principle alternatives for decoding. The first option is to use the Markov dependencies directly as further links between variable nodes in the Tanner graph of the LDPC code. The second alternative is to see the Markov source and the LDPC code as a serial concatenation asking for a Turbo iterative decoding between the two corresponding decoders. The latter comes with significant complexity compared to the direct embedding into the LDPC Tanner graph. Especially, one modification applying Jensen's inequality leads to a very low decoding complexity at no performance loss. The Turbo scheme's performance depends on the scheduling. Superior performance can be achieved with sufficient iterations in the LDPC decoder itself.**

I. Introduction

Since the mid 90s, LDPC codes have seen a resurgence after their initial discovery in 1963 by Gallagher. Properties of LDPC codes such as their capacity approaching behavior and low complexity decoding, coupled with the computational complexity scaling linearly with increasing block length, make them an excellent choice for many applications in different communications and data storage systems. LDPC codes are recommended as part of the digital video broadcasting version-2 (DVB-2) standard, as well as part of the Wi-Fi 802.11 standard, and used for 10GBase-T Ethernet, LTE etc. For this paper, the belief-propagation decoding of LDPC codes was modified to incorporate redundancy emanating from memory properties of the source sequence. Many sources in real life can be modeled as being generated from a Markov model. Here, we include the underlying model into the decoding process via the a-priori information.

Source and channel coding are done in a tandem fashion for most applications due to Shannon's separation principle[1]. For practical implementation, joint source-channel (JSC) coding methods are investigated for obtaining gains in fidelity and complexity [2], for example, in case of transmission in the non-asymptotic region where constraints in block length and decoder complexity exist. In [3], a joint design employing LDPC codes was presented, and the optimization of the design investigated. It was shown that the proposed design offered a significant improvement over existing JSC systems with LDPC codes. The proposed design combined two LDPC codes, used for source and channel coding and presented a compound graph with inter-connections which offered an improvement over the original design of such serial concatenated joint source and channel codes done using LDPC codes, presented in [4], [5].

In this paper, we investigate using the memory of the source sequence in decoding. For source sequences emanating from a Markov model, the belief propagation decoding of LDPC codes are modified to incorporate the Markov dependency. This modification can be done via a direct left-to-right link along the variable nodes, representing the dependency as an a-priori LLR value. An alternative method uses a BCJR decoder for estimating the Markov sequence, resulting in a turbo-like decoding scheme where the inner LDPC belief propagation decoding and the outer BCJR decoder exchange information iteratively. We provide simulation results confirming the superior performance of the modified approaches over not taking the memory into consideration.

The paper is structured as follows: in Section II, we provide a system description. In sections III and IV, we present two different methods of incorporating an underlying Markov model into the LDPC decoding process and also discuss the computational complexity. We present results in Section V and conclude by summarizing and outlining future steps in Section VI.

II. System Description

The JSC system proposed in [3], is a modification of a compound Tanner graph, first introduced in [4] and investigated for a JSC application in [5]. The system consists of two serially concatenated LDPC codes, the first of which is used for syndrome-source compression and the second as a channel code. The decoding was done via message passing decoding on the two graphs. In order to mitigate the high error floors exhibited by the system, modifications such as improved connection profiles between the graphs as well as shortening were employed, in [3] and [6], which significantly improve performance.

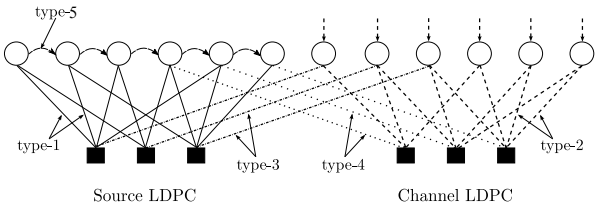

Fig. 1: JSC scheme with Markov links

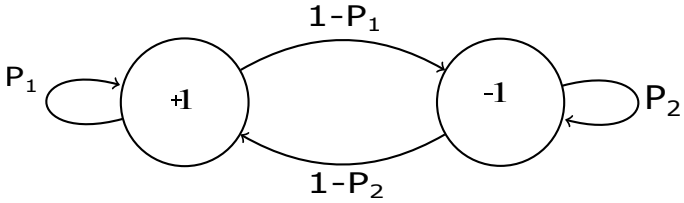

Fig. 2: First-order Markov source

As mentioned previously, we investigate LDPC decoding to include memory present in the source sequence resulting from a Markov model. This is done as a first step with an eventual JSC application in mind, illustrated in Fig. 1 and is a further modification of the systems described in [3], [6]. The exact structure shown is not yet followed in this paper. The design, optimization, and performance analysis for a final JSC application are ongoing. As a first step, here we consider the information bit sequence of a systematic LDPC code being generated by a Markov model; thereby already studying the situation that only the information variable nodes, i.e., a part of the codeword, show Markov relations. The goal of the paper is to investigate the LLR forwarding on a Tanner graph to incorporate memory properties of an uncompressed source, as represented in the left side of Fig. 1. A decoding scheme for such an LDPC - JSC scheme was briefly presented in [7]. Here, we proceed further with the decoding by introducing a computationally simpler version as well as a turbo-like decoding scheme. There are two possible approaches for decoding received analog values emanating from the model mentioned. In the first approach, as detailed in Section III, as well as the method mentioned in [7], additional to the received corrupted analog sequence, we use the Markov property of the source-sequence. The Markov transition probabilities are added as further inter-connections to the LDPC Tanner graph. In Section III, the decoding algorithm for this approach is presented as well as a further simplification. In Section IV, a hidden Markov model is investigated and subsequently, a turbo like scheme is employed to estimate the a-priori values of the information sequence.

III. Decoding Algorithm Considering Markov Source Properties

On the transmitter side, the binary information sequence $\mathbf{u}$ is generated by a Markov model shown in Fig. 2, with state transition matrix,

$$\mathbf{T} = \left[\begin{array}{c|cc} & +1 & -1 \\ \hline +1| & P_1 & 1-P_1 \\ -1| & 1-P_2 & P_2 \end{array}\right]$$

Since the source is an order-1 Markov state machine, there is a directed dependency between the variable nodes from left to right as shown with connection arrows labeled edge-type 5 in Fig. 1. A Markov sequence is generated by an auto-regressive state machine, i.e., the output of the previous state becomes part of next state. For order-1 models, the output becomes the next state directly. From the state transition matrix $\mathbf{T}$, a state-to-output transition matrix can be calculated, which for this simple case, is the same.

Decoding of LDPC codes is done using a Belief Propagation (BP) algorithm known as the Sum-Product algorithm (SPA), by passing LLR values along the edges of the Tanner graph. Three independent LLRs are combined to estimate the received sequence, bit wise, since the LDPC decoder is a MAP decoder. The following derivations are done using LLR values. An LLR value for u_q ,the q^{th} bit of the length k information sequence $\mathbf{u}$ is computed as $L(u_q) = \ln\frac{P(u_q=+1)}{P(u_q=-1)}$. The signs $+$ and $-$ of the LLR indicate output values $+1$ and -1 respectively, while the magnitude signifies the confidence of the estimate.

We assume transmission over an AWGN channel, the received analog values at the decoder are y, the transmitted codeword, $\mathbf{x}$ is of length n. Hence, the rate of the code is $R = \frac{k}{n}$.

The decision log-likelihood ratio at a node consists of three summands,

$$L = L_{intrinsic} + L_{a-priori} + L_{extrinsic}$$

The extrinsic information is computed by message passing on the Tanner graph by fulfilling the parity check constraints of the code. Since the Markov property is a characteristic of the source sequence, it is a-priori information. Hence, we improve a-priori estimate of the information bits by using the Markov source properties. Since we assume an order-1 source, the dependency only extends to the following bit to the right. The Markov model parameters, steady state distribution and state transition matrix are assumed to be available at the decoder.

The a-priori information can be viewed as an incoming contribution from the left node to the right, at the first k variable nodes, due to the systematic code construction. We provide two different formulae, where the second is an approximation with low complexity for estimating Markov a-priori information. Since we take a Markov sequence view of the received analog sequence, keeping in mind the order-1 model, the probability for the bit value of the node to the right can be computed for all information nodes, using $\mathbf{T}$. This estimate is used as the a-priori information, the derivation is provided for a node indexed by q with left node $q-1$ and $q \neq 0$.

$$
\begin{aligned}
L(u_q) &= \ln\left[\frac{P(u_q=+1)}{P(u_q=-1)}\right] \\
&= \ln\left[\frac{P(u_q=+1|u_{q-1}=+1)\cdot P(u_{q-1}=+1)+P(u_q=+1|u_{q-1}=-1)\cdot P(u_{q-1}=-1)}{P(u_q=-1|u_{q-1}=+1)\cdot P(u_{q-1}=+1)+P(u_q=-1|u_{q-1}=-1)\cdot P(u_{q-1}=-1)}\right] \qquad (1)\\
&= \ln\left[\frac{P(u_q=+1|u_{q-1}=+1)\cdot \frac{P(u_{q-1}=+1)}{P(u_{q-1}=-1)}+P(u_q=+1|u_{q-1}=-1)}{P(u_q=-1|u_{q-1}=+1)\cdot \frac{P(u_{q-1}=+1)}{P(u_{q-1}=-1}+P(u_q=-1|u_{q-1}=-1)}\right] \\
&= \ln\left[\frac{P_1\cdot\exp(L(u_{q-1}))+(1-P_2)}{(1-P_1)\cdot\exp(L(u_{q-1}))+P_2}\right] \qquad (2)
\end{aligned}
$$

In (2), the term $L(u_{q-1})$ refers to the decision LLR of the previous node, i.e., it is the sum of all messages on all incoming edges to the previous node, i.e., it is the current estimate of bit $q-1$ in our sequence view. Hence, the outgoing message on edge i of the variable node q at iteration l is,

$$
L(v_{q,i\rightarrow})^{(l)} = L(y_q|x_q)^{(l)} + \sum_{j,j\neq i} L(c_j)^{(l)} + \ln\left(\frac{P_1\cdot\exp(L(u_{q-1})^{(l-1)})+(1-P_2)}{(1-P_1)\cdot\exp(L(u_{q-1})^{(l-1)})+P_2}\right).
$$

where $L(c_j)$ are incoming messages from the check nodes.

After every iteration of the LDPC decoder, the current estimates for the information sequence are calculated and stored. For the next iteration, we use the estimate from the previous iteration, piecewise, to provide an a-priori value for each bit of the information sequence, from left to right.

We assume blocked transmission of the information sequence which is generated continuously. Hence, for the first bit of every codeword, since there is no left node, the a-priori probability is taken as the ratio of the steady state probabilities. Similarly, for the first iteration when no estimate of the received sequence is available, all nodes are initiated with a-priori probability as the ratio of the steady states,

$$
L_{\text{a-priori-init}} = \ln\left(\frac{P(L(u_0)=+1)}{P(L(u_0)=-1)}\right) = \ln\left(\frac{P_s(+1)}{P_s(-1)}\right) \qquad (3)
$$

In (1), the numerator is computed by summing the probabilities for each state in the model to generate a $+1$. The denominator sums the probabilities of all states to generate a -1. Thus, if the Markov model has m states, the numerator and denominator both will have m summands. The probabilities for the states to generate a $+1$ or -1 are weighed by the probability to be in that state, which can be calculated from the previous m bits of the received sequence, due to the auto regressive structure of a Markov state machine. The decoding easily generalizes for arbitrary order-m Markov models. In the simulation results provided in Section V, a performance curve for an order-2 model is provided.

A. Simplified Computation

Equation (2) uses the outgoing probabilities from a state to compute the probability of the next state. This is a 'predictive' approach. We present an additional method for computing the a-priori values. From Fig. 2 we calculate LLRs at the states for generating outputs; where $st=+1$ and $st=-1$ denote the states.

$$
L_{(st=+1)} = \ln\frac{(+1|st=+1)}{(-1|st=+1)} = \ln\left[\frac{P_1}{1-P_1}\right], \qquad (4)
$$

$$
L_{(st=-1)} = \ln\frac{(+1|st=-1)}{(-1|st=-1)} = \ln\left[\frac{1-P_2}{P_2}\right]. \qquad (5)
$$

The current estimates at node $q-1$, $P(u_{q-1})=+1$ and $P(u_{q-1})=-1$, are respectively represented by, $P(+)$ and $P(-)$. We scale the estimates with (4) and (5) and obtain the LLR for the next bit.

$$
\begin{aligned}
L(u_q) &= P(+)\cdot\ln\left[\frac{P_1}{1-P_1}\right]+P(-)\cdot\ln\left[\frac{1-P_2}{P_2}\right], \qquad (6)\\
&= \underbrace{P(+)\ln(P_1)+P(-)\ln(1-P_2)}_{a} \\
&\quad \underbrace{-P(+)\ln(1-P_1)-P(-)\ln(P_2)}_{b}. \qquad (7)
\end{aligned}
$$

The derivation presented in Section III and in (6) are related by Jensen's inequality, which states that, if f is a concave function and X is a random variable where $\sum p_i = 1$, then

$$
p_1\cdot f(x_1)+p_2\cdot f(x_2) \leq f(p_1\cdot x_1+p_2\cdot x_2) \qquad (8)
$$

From (1),

$$
\begin{aligned}
L(u_q) = &\underbrace{\ln\left[P(+)P_1+P(-)(1-P_2)\right]}_{c} \\
&\underbrace{-\ln\left[P(+)(1-P_1)+P(-)P_2\right]}_{d}
\end{aligned} \qquad (9)
$$

Since $\ln$ is a concave function, using (8),

$$
a \leq c; \quad b \geq d
$$

Since, the two summands of (7) are bounded respectively lower and higher, with respect to the two summands of (9), the overall effect of the approximation compensates. We will observe this from the simulation results, too.

$$
\begin{aligned}
&\ln\left[P(+)P_1+P(-)(1-P_2)\right]-\ln\left[P(+)(1-P_1)+P(-)P_2\right] \\
&\qquad \approx P(+)\cdot\ln\left[\frac{P_1}{1-P_1}\right]+P(-)\cdot\ln\left[\frac{1-P_2}{P_2}\right]. \qquad (10)
\end{aligned}
$$

This method is computationally simpler, as $\ln[\frac{P_1}{1-P_1}]$ and $\ln[\frac{1-P_2}{P_2}]$ are fixed values. Hence, the expression is linear in $P(+)$ or $P(-)$ and as shown in the simulation results, provides similar performance.

IV. Hidden Markov Model View

A hidden Markov model is characterized by a state transition matrix and an emission matrix. In HMM literature, a classic problem is to determine the state sequence of the encoder, given the observed output sequence. Both Viterbi and BCJR decoders are used for this task. In this section, we reformulate our system as an HMM.

On the decoder side, the received analog values are the output of the Markov state machine corrupted by additive i.i.d. Gaussian noise with zero mean and standard deviation σ. We consider this observed sequence to be the continuous output of an HMM model, as shown in Fig. 3. The channel statistics are viewed as continuous emission probability densities.

Since the decoder only observes the continuous outputs, and has knowledge of the channel statistics as well as $\mathbf{T}$, it can be viewed as a classical problem of finding the sequence of states of the system.

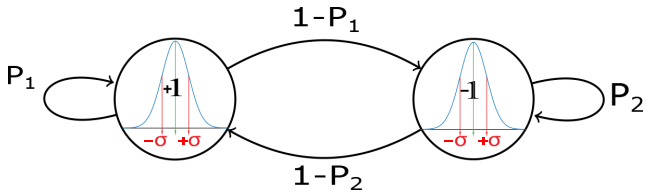

Fig. 3: Hidden Markov model

To this end, we construct a decoder architecture similar to a serial concatenated Turbo decoding scheme. Our goal is to use the BCJR algorithm to deduce the probable state sequence and use this a-priori information in the LDPC decoder. The LDPC code performs parity-checks and provides a-priori information for the BCJR decoder. The system is described in Fig. 4. The outer Markov state machine block produces a continuous stream of outputs which is blocked, $k - bits$ at a time for transmission. The LDPC encoder generates codewords $\mathbf{x}$ of length n. After transmission, on the decoder side, the intrinsic information of the received bits is fed into the LDPC decoder, along with the a-priori estimate, computed by the BCJR decoder. One iteration of the decoder is counted as serial decoding performed by the LDPC decoder first and subsequently, by the BCJR decoder. In the first iteration, there is no incoming information from the BCJR decoder, hence $L_{\text{ap}}(i_k) = 0$. The output of the LDPC decoder, provides a-priori information for the BCJR decoder, after subtracting the input the BCJR decoder provided in that iteration, to avoid information looping, and vice-versa for the LDPC decoder.

The information exchange on the concatenated graph is shown in Fig. 5. In this serial concatenation scheme, each decoder provides a-priori estimates for the other. The Trellis provided here is of a 2-state system, i.e., order-1. For order-1, the outputs of the Markov model become the states for the next

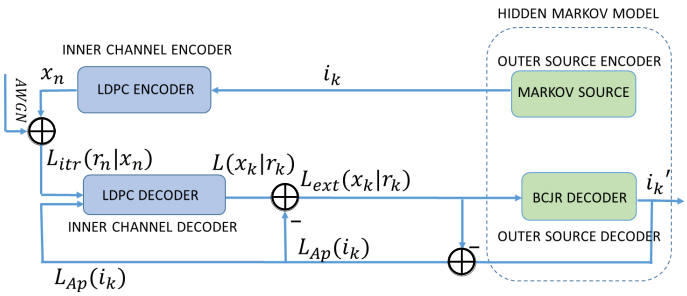

Fig. 4: Concatenated encoder and decoder structure

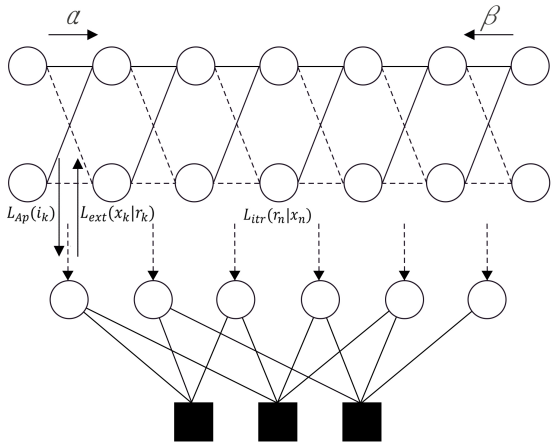

Fig. 5: Information exchange between the LDPC and BCJR decoders

time instant. We start our iterations in the inner LDPC decoder which estimates the value of the information sequence at the variable nodes. These estimates map onto the state transitions in the Trellis. Solid lines represent an output +1 and dashed lines represent an output -1. The probabilities of being +1 and -1 are computed from the outgoing $L_{ext}(x_k|r_k)$ of the LDPC decoder and placed on the corresponding Trellis paths as shown by the dashed arrow from the variable nodes to the state transitions. The BCJR decoder computes three probability estimates, α and β values are the probabilities to be in the states considering all paths converging to and stemming from the states, respectively. γ is computed as a product of the state transitions on the Trellis given by $\mathbf{T}$ and $L_{ext}(x_k|r_k)$ values provided by the LDPC decoder. The BCJR computes a MAP estimate for every output bit as a product of these independent estimates, which provides a-priori values for the LDPC decoder, as shown by the solid curved lines from the BCJR to the variable nodes.

V. Performance Comparison

In this section we provide some simulation results. An irregular rate-$1/2$ LDPC code was constructed using the following variable- and check-node degree distribution polynomials,

$$\lambda(x) = 0.28286x + 0.39943x^2 + 0.31771x^7,$$
$$\rho(x) = 0.6x^5 + 0.4x^6.$$

The PEG algorithm was used to construct a systematic triangular $\mathbf{H}$ matrix. BER results were compiled after atleast 100 independent errorred words were received. The BER curves are presented in Fig. 6. For ease of reference, the decoding algorithms presented in Section III, III-A, and IV will be referred to as, Dec-1, Dec-2, and Dec-Ser. As reference, performance curves of an LDPC decoder using the sum-product algorithm are provided, shown in blue and green, labeled Ref T_1 and Ref T_2. These curves are generated for $\mathbf{T}$ matrices, respectively,

$$\mathbf{T_1} = \begin{pmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{pmatrix}, \quad \mathbf{T_2} = \begin{pmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{pmatrix}.$$

As expected, the curves lie on top of each other since LDPC codes guarantee performance independent of the individual codewords. We then plot the results for Dec-1. The underlying transition matrices were $\mathbf{T_1}$ and additionally,

$$\mathbf{T_3} = \begin{pmatrix} 0.6 & 0.4 \\ 0.4 & 0.6 \end{pmatrix}, \quad \mathbf{T_4} = \begin{pmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{pmatrix}.$$

We observe from the results, incorporating memory into decoding improves the performance significantly. The higher the memory within the sequence, the better the performance since the a-priori estimates for sequences with stronger memory are more reliable. Results for Dec-2 are plotted for $\mathbf{T_1}$. We observe that the performance curves computed by the two methods are more or less identical. Results for the order-2 (O:2) Markov model are also provided, for $\mathbf{T_5}$ and Dec-1.

$$\mathbf{T_5} = \left[\begin{array}{c|cccc} & (+1,+1) & (1,-1) & (-1,1) & (-1,-1) \\ \hline (+1+1)| & 0.9 & 0 & 0.1 & 0 \\ (+1-1)| & 0.5 & 0 & 0.5 & 0 \\ (-1+1)| & 0 & 0.5 & 0 & 0.5 \\ (-1-1)| & 0 & 0.1 & 0 & 0.9 \end{array}\right]$$

For the order-2 model, we compare the results obtained with decoding the sequence by only taking into account order-1 memory. The order-1 equivalent matrix of $\mathbf{T_5}$ is $\mathbf{T_4}$, obtained by marginalizing over the right-most bit of the states, in both rows and columns. We again observe that there is a performance loss when the full memory of the sequence is not taken into account.

For the turbo-like decoding scheme, scheduling is an important issue. For our case, from preliminary results, we observe that a one-pass serial decoding schedule does not provide any gain over the other two methods described. This is due to the fact that, with each call of the LDPC decoder, only one cycle of message passing is performed in the Tanner graph. LDPC-like belief propagation decoding methods provide improved estimates due to their iterative procedure, the benefits of which cannot be exploited in this serial decoder structure. To make use of the iterative message passing within the Tanner graph, the one-pass serial decoding was performed 10 times. The resultant estimate of the a-priori LLR from the BCJR decoder, $L_{\text{ap}}(i_k)$ was then used in a regular LDPC decoder; which does not consider the memory of the sequence; and the BER was computed. The total number of iterations in this scheme is 20. In order to have a fair comparison, Dec-1 was called with 20 iterations for the same Markov model $\mathbf{T_1}$, labeled

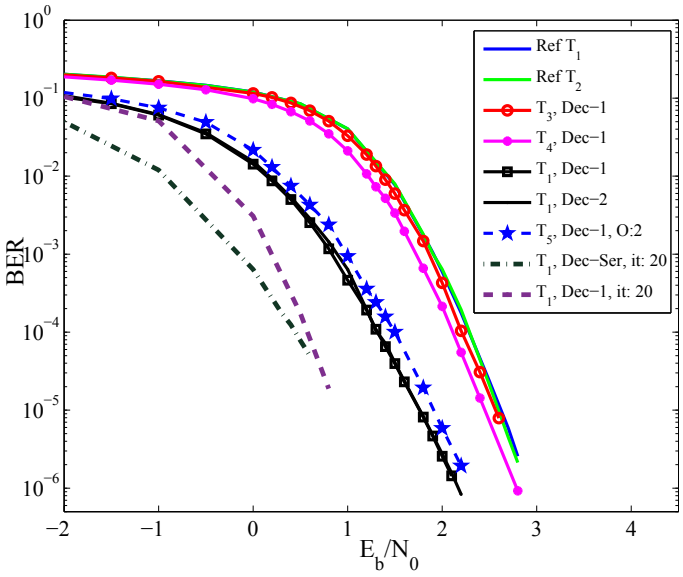

Fig. 6: Simulation results for Dec-1, Dec-2, and Dec-Ser for different $\mathbf{T}$ matrices

$T_1, Dec-1, it:20$. We observe that the serial architecture performs better.

VI. Conclusion

Modifying message passing decoding to include memory properties of the source sequence improve the performance of LDPC codes. The design of LDPC codes for these modified decoding methods is a future step for this work. The scheduling for the concatenated decoder will provide further insight into the applicability of the computationally complex concatenated model over the other two methods with very low complexity.

Acknowledgment

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – HE 3654/16-1.

References

[1] C. E. Shannon, "A mathematical theory of communication," *Bell Systems Techn. Journal*, vol. 27, pp. 623–656, 1948.

[2] J. L. Massey, "Joint source and channel coding," *Communications Systems and Random Process Theory*, vol. 11, pp. 279–293, Sijthoff and Nordhoff, 1978.

[3] H. V. Beltrão Neto and W. Henkel, "Multi-edge optimization of low-density parity-check codes for joint source-channel coding," in *Proc. of 9th International ITG conference on Systems, Communications and Coding*, Munich, Germany, Jan. 2013.

[4] G. Caire, S. Shamai, and S. Verdú, "Almost-noiseless joint source-channel coding-decoding of sources with memory," in *Proc. 5th International ITG Conference on Source and Channel Coding*, Jan. 2004, pp. 295–304.

[5] M. Fresia, F. Pérez-Cruz, and H. V. Poor, "Optimized concatenated LDPC codes for joint source-channel coding," in *Proc. IEEE Int. Symposium on Information Theory*, Seoul, South Korea, 2009.

[6] H. V. Beltrão Neto and W. Henkel, "Information shortening for joint source-channel coding schemes based on low-density parity-check codes," in *Proc. of 8th International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, Bremen, Germany, August 2014.

[7] Z. Mei and L. Wu, "LDPC codes for binary markov sources." [Online]. Available: www.paper.edu.cn