Memory-Assisted Quantized LDPC Decoding

Philipp Mohr, *Student Member, IEEE*, and Gerhard Bauch, *Fellow, IEEE*

***Abstract*—This letter significantly improves coarsely quantized decoding of 5G LDPC codes up to 0.36 dB by reusing computed check node messages from previous iterations. Typically, variable and check nodes generate and replace old messages in every iteration. We show that, under coarse quantization, discarding old messages involves a significant mutual information loss that is avoided with additional memory. We propose a modified information bottleneck algorithm to design node operations taking messages from the previous iteration(s) into account as side information. Finally, we reveal a 2-bit row-layered decoder that can operate within 0.25 dB w.r.t. 32-bit belief propagation.**

***Index Terms*—LDPC decoder, layered decoding, rate-compatible, coarse quantization, information bottleneck**

I. Introduction

EFFICIENT and reliable decoding of low-density parity-check (LDPC) codes is vital in modern technologies with high data rate requirements, such as 5G [1]. Particularly the exchange of messages in iterative message passing decoding algorithms like belief propagation demands significant complexity [2]. To overcome this bottleneck many works focus on reducing the bit width of the exchanged messages in these algorithms through quantization operations [3]–[11].

The quantized messages typically encode reliability information exchanged between variable nodes (VNs) and check nodes (CNs). The choice of reliability levels is crucial for excellent decoding performance with low-resolution messages [11]. Information optimum reliability levels can be found with the information bottleneck (IB) method which is a clustering framework, that enables the design of compression operations for maximizing preserved relevant information [8]–[13]. Relevant mutual information measures the average amount of information between the transmitted code bits and exchanged decoding messages.

Typically, calculated messages from a previous iteration are replaced by updated messages from the current iteration [3]–[11]. One might question whether discarding previously computed and exchanged messages wastes valuable information. Indeed, under coarse quantization, this work confirms that preserving old messages of the previous iteration can significantly improve the decoding performance. For the design of a memory-assisted decoder we modify the sequential IB algorithm from [8] to be aware of the messages retained in memory. This algorithm is specifically suited for the design of deterministic compression mappings realized with symmetric thresholds. It has significantly reduced computational costs compared to more general solutions [13].

We combine the memory-assisted decoder structure with our recently proposed region-specific CN-aware quantizer design.

Philipp Mohr and Gerhard Bauch are with the Institute of Communications, Hamburg University of Technology, Hamburg, 21073, Germany. E-mail: {philipp.mohr; bauch}@tuhh.de.

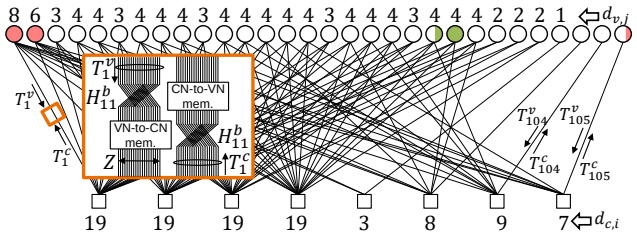


Fig. 1: Tanner graph of a 5G LDPC code [11].

Region-specific quantization allows individual alphabets of reliability levels for subsets of exchanged messages particularly improving low-resolution decoding of highly irregular 5G-LDPC codes [11]. A CN-aware quantizer design for the VN extends the optimization scope to maximize preserved relevant information at the output of the subsequent CN update [10], [11]. The combination of this work and [11] yields up to 0.68 dB gain w.r.t. 2-bit decoding without those techniques.

II. Preliminaries on LDPC Decoding with Mutual Information Maximizing Quantization

Most standards define LDPC codes through a base matrix $\mathbf{H}_b$ with entries $H_{ij}^b \in \{-1, \dots, Z\}$. The base matrix $\mathbf{H}_b$ can be represented by a Tanner graph illustrated in Fig. 1. Each column j turns into a variable node (VN) and each row i into a check node (CN). The non-negative entries H_{ij}^b are edges between VNs and CNs. The node degree, i.e., the number of connected edges to a node, is $d_{v,j}$ for a VN and $d_{c,i}$ for a CN.

Lifting replaces every edge with Z edges that are subjected to H_{ij}^b-cyclic permutation. The lifted graph can be equivalently represented by a lifted parity check matrix $\mathbf{H}$. The encoder maps the information bits $\boldsymbol{u}$ to code bits $\boldsymbol{b}$ such that $\mathbf{H}\boldsymbol{b}=\mathbf{0}$ [11]. The decoder in the receiver assumes a memoryless channel. For every $b \in \boldsymbol{b}$ a binary channel LLR L^{ch} is quantized to a w^{ch}-bit message t^{ch} as input to the decoder. The quantization maximizes mutual information between b and t^{ch} as in [11].

A. Decoding with Arbitrary Schedules

Message passing decoding computes and exchanges messages between VNs and CNs to aggregate soft information for error correction from the parity check constraints. Each edge of the graph contains a VN and CN memory location enumerated with $m \in \mathcal{N} = \{1, \dots, \sum_j d_{v,j}\}$ for the VN-to-CN messages and $n \in \mathcal{N}$ for the CN-to-VN messages (cf. Fig.1). A memory location stores Z messages after lifting the graph as illustrated within the orange box in Fig. 1. The sets $\mathcal{U}^v \subseteq \mathcal{N}$ and $\mathcal{U}^c \subseteq \mathcal{N}$ specify target memory locations for VN and CN updates, respectively. The decoding schedule defines the order in which memory locations are updated as

$$\boldsymbol{\mathcal{U}} = (\mathcal{U}_0^v, \mathcal{U}_1^c, \mathcal{U}_2^v, \mathcal{U}_3^c, \dots) \tag{1}$$

followed by a final hard decision update that uses the most recent updated CN messages.

B. Node Operations

We introduce the discrete random variables T_j^{ch}, T_m^v and T_n^c for modeling the channel, VN and CN messages. A realization $t_\triangleright^\diamond$ of $T_\triangleright^\diamond$ takes values from an LLR-sorted alphabet $\mathcal{T}_\triangleright^\diamond=\{\pm1,..,\pm2^{w_\diamond-1}\}=\{-2^{w_\diamond-1},..,-1,1,..,2^{w_\diamond-1}\}$ where $\diamond\in\{ch,v,c\}$ and $\triangleright\in\{j,m,n\}$. We set $w_{ch}=5$ and $w_v=w_c=w$ in this paper. For the design of the decoder, we keep track of $p(x_m,t_m^v)$ and $p(x_n,t_n^c)$ that change with every VN and CN update. Updating a VN memory location $m\in\mathcal{U}^v$ yields [11]

$$y_m^v = L(x_m|t_{\mathrm{col}(m)}^{ch}) + \sum_{n\in\xi_m^v} \bar{L}(t_n^c|x_m) \tag{2}$$

with extrinsic CN locations $\xi_m^v=(n{:}n\neq m,\mathrm{col}(n)=\mathrm{col}(m))$. An LLR reconstruction of a message t_n^c is denoted as $\bar{L}(t_n^c|x)=\log p(\bar{T}_n^c=t_n^c|x=0)/p(\bar{T}_n^c=t_n^c|x=1)$ using the aligned variables $\bar{T}_n^c$ introduced in the next subsection II-C. The hard decision yields $\hat{x}_m=(1-\mathrm{sgn}(\hat{L}_m))/2$ with the a-posteriori probability (APP) LLR $\hat{L}_m=y_m^v+\bar{L}(t_m^c|x_m)$. Row-layered decoder structures, such as the one in Fig. 8, typically use the APP LLR for computing (2) efficiently as $y_m^v=\hat{L}_m-\bar{L}(t_m^c|x_m)$. Those decoders initialize the APP LLR with $\hat{L}_m=L(x_m|t_{\mathrm{col}(m)}^{ch})$. Hence, the reconstruction of the channel message must be done only once for all iterations. In an implementation the reconstruction in (2) is typically carried out with integer scaled LLRs of bit width $w'\approx8$ to avoid performance loss [11].

Updating a CN memory location $n\in\mathcal{U}^c$ yields [3] [11]

$$t_n^c = Q(y_n^c) \text{ with } y_n^c = \prod_{m\in\xi_n^c} \mathrm{sgn}(y_m^v) \min_{m\in\xi_n^c} |y_m^v| \tag{3}$$

with extrinsic VN locations $\xi_n^c=(m{:}m\neq n,\mathrm{row}(m)=\mathrm{row}(n))$. Without performance loss, the quantizer Q can be moved before the CN update (cf. Fig. 2a & 2b) lowering complexity.

C. Alignment Regions

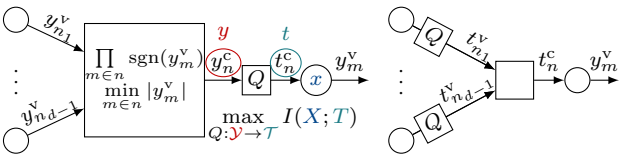
To reduce design and implementation complexity, an alignment operation is applied to the variables T_n^c and X_n *before* the node design as

$$\bar{T}_n^c = \frac{1}{|\mathcal{A}_n|}\sum_{n'\in\mathcal{A}_n} T_{n'}^c \text{ and } \bar{X}_n = \frac{1}{|\mathcal{A}_n|}\sum_{n'\in\mathcal{A}_n} X_{n'} \tag{4}$$

where $\mathcal{A}_n$ comprises all elements from the same region. This work considers a row-alignment $\mathcal{A}_n=\{n'{:}\,\mathrm{row}(n')=\mathrm{row}(n)\}$ or matrix-alignment $\mathcal{A}_n=\mathcal{N}$ [11]. All messages t_n^c in the same alignment region share the same reconstruction function $\bar{L}$ in (2) and quantization function Q in (3).

D. Region-Specific Quantization with Check Node Awareness

A compact version of the CN messages can be obtained with threshold quantization $t_n^c=Q(y_n^c)$ (cf. Fig. 2a). The objective is to maximize the mutual information $\max_Q I(\bar{X}_n;\bar{T}_n^c)$ preserved by any CN message in the alignment region $\mathcal{A}_n$. The optimization of Q can be performed with the sequential IB algorithm [8], also described in section IV. This algorithm requires the distribution $p(\bar{x}_n,\bar{y}_n^c)=\sum_{n'\in\mathcal{A}_n} p(x_{n'},Y_{n'}^c=\bar{y}_n^c)/|\mathcal{A}_n|$. In case of a layered schedule, each update points to a subset of all memory locations $\mathcal{U}\subset\mathcal{N}$. *A single* quantizer design must only be performed for each region $\mathcal{N}_a\in\{\mathcal{A}_n{:}n\in\mathcal{U}\}$ where $a\in\{1,\ldots,|\mathcal{A}|\}$ enumerates the distinct regions. The notation $\{\mathcal{A}_n{:}n\in\mathcal{U}\}$ builds a set with unique elements, e.g., $\{\{1,2\},\{2,1\},\ldots\}$ reduces to $\{\{1,2\},\ldots\}$. The quantizer designed for region $\mathcal{N}_a$ is used only for updating the locations defined by $\mathcal{N}_a\cap\mathcal{U}$ [11].



(a) CN-aware design of Q. (b) Q is implemented before CN.

Fig. 2: Design and implementation of CN-aware quantization.

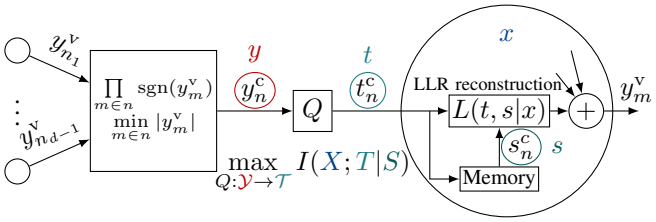


Fig. 3: Quantizer design with memory-assisted reconstruction.

III. Memory-Assisted Reconstruction

A small bit width w of the exchanged messages significantly lowers the decoder complexity for several reasons:

- One quantization operation uses $w-1$ comparisons [10].
- The routing network size scales with w.
- The min-CN update can be carried out much faster. For example using $w=2$ instead of $w=3$ bits potentially reduces the logic gate delay by a factor of 4 [10].

Unfortunately, with $w=2$ bit these major complexity savings can lead to a noticeable performance degradation [11]. This section proposes a novel approach to overcome most of the degradation when using e.g. $w=2$ instead of $w=3$ bits. Fig. 3 extends the setup by preserving the old CN message t_n^c of the previous iteration as s_n^c. We remark that preserving s_n^c for another iteration would give only marginal performance gains.

A. Modification of Existing Decoder Design

Instead of aiming for $\max_Q I(\bar{X};\bar{T}_n^c)$ we propose to take the statistics of the message s_n^c into account. The message s_n^c already provides a certain amount of mutual information $I(X_n^c;S_n^c)$. The optimization of the quantizer shall optimize preservation of additional mutual information $\max_Q I(X_n^c;T_n^c|S_n^c)$. The variable node update now takes into account s_n^c as

$$y_m^v = L(x_m|t_{\mathrm{col}(m)}^{ch}) + \sum_{n\in\xi_m^v} \bar{L}(t_n^c,s_n^c|x_m) \tag{5}$$

For the design of Q we measure the joint distribution $p(x_m,y_m^c,s_m^c)$. For the measurement we generate a large set of decoding messages x_m,y_m^c,s_m^c under a specific design-E_b/N_0. In the next section IV we introduce a side-information aware IB algorithm which is used for optimization of the quantization thresholds. We define X_n, Z_n^c, T_n^c, and S_n^c as the

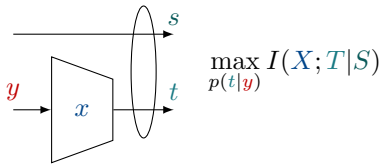


Fig. 4: Information bottleneck setup with side information. The mapping $p(t|y)$ is realized with threshold quantization.

relevant, observed, compressed and side-information variable, X, Y, T, and S, respectively. The optimization aims for $\max_{p(t|y)} I(X;T|S)$. We remark that the alphabet $\mathcal{Y}$ is not strictly LLR-sorted as defined in (7) as a result of the CN minimum approximation.

IV. Information Bottleneck Algorithm with Side-Information Awareness

Typically, an IB setup is defined by a relevant, observed and compressed random discrete variable $X, x{\in}\mathcal{X}{=}\{0,1\}$, $Y, y{\in}\mathcal{Y}{\in}\{1,\ldots,|\mathcal{Y}|\}$ and $T, t{\in}\mathcal{T}{=}\{1,\ldots,|\mathcal{T}|\}$ that form a Markov chain $X{\to}Y{\to}T$. The IB method is a generic clustering framework for designing compression operations $p(t|y)$ with optimization objective $\max_{p(t|y)} I(X;T) - \beta^{-1} I(Y;T)$. The choice of $\beta \geq 0$ allows to trade preservation of relevant information $I(X;T)$ for compression. Of very high practical interest is the case where $\beta{\to}\infty$ because it can be achieved with a deterministic mapping through threshold quantization

$$t = Q(y) = \mathcal{T}[k] \quad \tau_k{\leq}y{<}\tau_{k+1}, 0{\leq}k{<}|\mathcal{T}| \tag{6}$$

with outer thresholds $\tau_0{=}0$ and $\tau_{|\mathcal{T}|}{=}|\mathcal{Y}|$, and $\mathcal{T}[k]$ identifying the kth element of the ordered set $\mathcal{T}$. The mapping with thresholds can be information-optimum if [14]

$$L(x|y{=}1){\leq}L(x|y{=}2){\leq}\ldots{\leq}L(x|y{=}|\mathcal{Y}|). \tag{7}$$

This section extends the conventional setup with a fourth variable S which provides side-information about X [13]. The setup is depicted in Fig. 4. Prior works [12], [13] also consider a setup with side information, however, those works do not explicitly provide a low-complexity solution for optimizing a threshold quantizer in the context of LDPC decoding. We propose Algorithm 1 which is a modified variant of the sequential IB algorithm from [8] taking into account the side information S. The algorithm exploits (7) to sequentially optimize initial random boundaries τ_k defining the target clusters $\mathcal{Y}_t{=}\{y{:}\tau_t{\leq}y{<}\tau_{t+1}\}{\subset}\mathcal{Y}$. We enforce symmetric thresholds $\tau_k{=}|\mathcal{T}|{-}\tau_{|\mathcal{T}|-k}$ for reducing complexity [8].

A. Merger Costs with Side Information

In line 14, the element y and counterpart element y' are moved into singleton clusters $\mathcal{Y}_{|\mathcal{T}|+1}$ and $\mathcal{Y}_{|\mathcal{T}|+2}$, respectively. The temporary decompression is modeled with a discrete random variable $\ddot{T}, \ddot{t}{\in}\ddot{\mathcal{T}}{=}\mathcal{T}{\cup}\{|\mathcal{T}|{+}1, |\mathcal{T}|{+}2\}$. Line 16 optimizes the deterministic mapping $p(t|\ddot{t}){=}\delta(t{-}f_k(\ddot{t}))$ with $f_k : \ddot{\mathcal{T}}{\to}\mathcal{T}$. The algorithm restricts merging y into an adjacent cluster t_1 or t_2. Thus, two mapping options exist $k{\in}\{1,2\}$ with

$$f_k(\ddot{t}){=}\begin{cases} \ddot{t} & \ddot{t}{\in}\mathcal{T} \\ t_k & \ddot{t}{=}|\mathcal{T}|{+}1 \\ t'_k & \ddot{t}{=}|\mathcal{T}|{+}2 \end{cases} \quad \text{with } t'_k{=}|\mathcal{T}|{+}1{-}t_k. \tag{9}$$

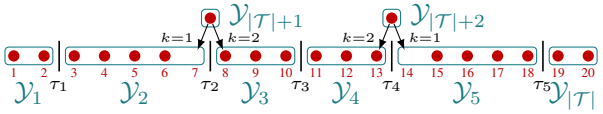
Algorithm 1 A sequential IB algorithm from [8] considering side information s in the merger cost computation (line 16).

Input: $p(x,y,s), |\mathcal{T}|$
Output: $p(t|y)$, $p(x,t,s)$

1: Create random symmetric clustering $p(t|y)$;
2: Compute $p(x,t,s) = \sum_y p(t|y)p(x,y,s)$;
3: Extend clusters $\mathcal{Y}_1,\ldots,\mathcal{Y}_{|\mathcal{T}|}$ with empty singleton clusters $\mathcal{Y}_{|\mathcal{T}|+1}$ and $\mathcal{Y}_{|\mathcal{T}|+2}$;
4: **repeat**
5: Save $p(t|y)$ as $p^{\text{old}}(t|y)$;
6: **for** $t \in \{1,\ldots,|\mathcal{T}|/2{-}1\}$ **do**
7: **for** $(t_1,t_2) \in \{(t,t{+}1),(t{+}1,t)\}$ **do**
8: **repeat**
9: **if** $t_1 == t$ **then**
10: y is rightmost element in cluster $\mathcal{Y}_{t_1}$;
11: **else if** $t_1 == t{+}1$ **then**
12: y is leftmost element in cluster $\mathcal{Y}_{t_1}$;
13: **end if**
14: Move y into $\mathcal{Y}_{|\mathcal{T}|+1}$, and $y'{=}|\mathcal{Y}|{+}1{-}y$ into $\mathcal{Y}_{|\mathcal{T}|+2}$;
15: Update $p(x,t,s)$;
16: Compute optimal cluster using (12):

$$k = \arg\min_{k^*\in\{1,2\}} C_{sym}(y,k^*) \tag{8}$$

Example: $|\mathcal{Y}|{=}20$, $|\mathcal{T}|{=}6$, $t{=}2$, $(t_1,t_2){=}(2,3)$, $y{=}7$, $y'{=}14$



17: Merge $\mathcal{Y}_{|\mathcal{T}|+1}$ into $\mathcal{Y}_{t_k}$, and $\mathcal{Y}_{|\mathcal{T}|+2}$ into $\mathcal{Y}_{|\mathcal{T}|+1-t_k}$;
18: Update $p(x,t,s)$ and $p(t|y)$;
19: **until** $t_k == t_1$
20: **end for**
21: **end for**
22: **until** $p(t|y) == p^{\text{old}}(t|y)$
23: **return** $p(t|y)$, $p(x,t,s)$

The mutual information loss from merging is

$$C_{sym}(y,k) = I(X;\ddot{T}|S) - I(X;T|S) \tag{10}$$

$$= \sum_{\ddot{t},s} p(\ddot{t},s)\, \mathrm{D}_{\mathrm{KL}}(p(x|\ddot{t},s)||p(x|f_k(\ddot{t}),s)) \tag{11}$$

$$= \sum_{s} p(s)(C(y,t_k|s){+}C(y',t'_k|s)) \tag{12}$$

where the individual merger costs are

$$\begin{aligned} C(y,t|s){=}p(\ddot{T}{=}t|s)\, \mathrm{D}_{\mathrm{KL}}\left\{p(x|\ddot{T}{=}t,s)||p(x|T{=}t,s)\right\} \\ {+}p(Y{=}y|s)\, \mathrm{D}_{\mathrm{KL}}\left\{p(x|Y{=}y,s)||p(x|T{=}t,s)\right\} \end{aligned} \tag{13}$$

with $\mathrm{D}_{\mathrm{KL}}(p||q){=}\sum_x p(x)\log_2(p(x)/q(x))$.

V. Evaluation with 5G Codes

This section investigates the performance of the proposed decoders with memory-assistance. As in [11] we use a 5G-LDPC code with length 8448, base graph 1 and various code rates [1]. Furthermore, we consider an AWGN channel with

BPSK modulation. All decoders use the initialization schedule described in [11] to avoid useless CN updates resulting from processing punctured messages. If not mentioned otherwise, the remaining schedule follows the flooding scheme with a maximum of 30 decoder iterations. Each decoder design uses a large set of training data with 10000 transmitted and received code words generated for a specific design E_b/N_0. Analytical tracking of joint probabilities for the design of memory-assisted decoders seems infeasible. For a fair comparison, also the conventional decoders are designed with the training data, leading to slightly different results compared to our work [11].

A. Evolution of Mutual Information

Fig. 5 depicts the evolution of mutual information between code bit X and the corresponding hard decision $\hat{X}$ for every iteration. The quantized messages are matrix-aligned such that all messages use the same alphabet of reliability levels in one iteration. The design process is initialized with the same design E_b/N_0. Particularly under 2-bit decoding, the mutual information gains per iteration are significantly improved with the proposed memory-assistance. For 3-bit decoding those gains appear smaller. Nevertheless, the proposed 3-bit decoding almost achieves the same performance as the conventional 4-bit decoding.

B. Boundary Placement for Memory-Assisted Reconstruction

This section analyzes the placement of quantizer boundaries τ_k for every iteration to explain the performance gains achieved with the proposed structure. Now, all decoders use an individual design E_b/N_0 so that the mutual information converges after 30 iterations $I(X,\hat{X})(30) \approx 0.9999$. A careful optimization is very important to ensure minimum frame error rate for a given budget of decoding iterations.

Fig. 6 shows that for all decoders the magnitude of boundaries increases for higher iterations as the reliability of messages improves. One key observation is that boundary magnitudes for the proposed 2-bit decoder show up an alternating rising and falling trend. This behavior clearly shows the effectiveness of the side-information aware IB algorithm: A CN message t is a compressed version of the non-quantized LLR y_n^c in (3) from the current iteration. A CN message s is a compressed version of $\underline{y}_n^c$ from the previous iteration. The difference $\Delta y_n^c = y_n^c - \underline{y}_n^c$ is sufficiently small on average, such that s can approximately resolve y_n^c. Different boundaries for s and t improve their combined capability to resolve y^c in *relevant ranges*. The combined resolution approaches the 3-bit quantizer in Fig. 6.

We remark that also without memory-aware optimization of the boundaries, significant performance gains can be achieved by performing the LLR reconstruction $L(t,s|x)$ with messages in memory. For example, consider a sign-magnitude alphabet sorted by underlying LLR with $t \in \mathcal{T} = \{-2,-1,+1,+2\}$. For $t=+2$ the reconstructed LLR $L(t=+2,s|x)$ is more reliable if the message in memory is matching, e.g., $s=+2$. It is less reliable if s does not agree, e.g., $s=-2$. Thus, $L(t=+2,s=+2|x) > L(t=+2,s=-2|x)$. Combined knowledge about the message s and t allows better inference of the non-quantized LLR y_n^c.

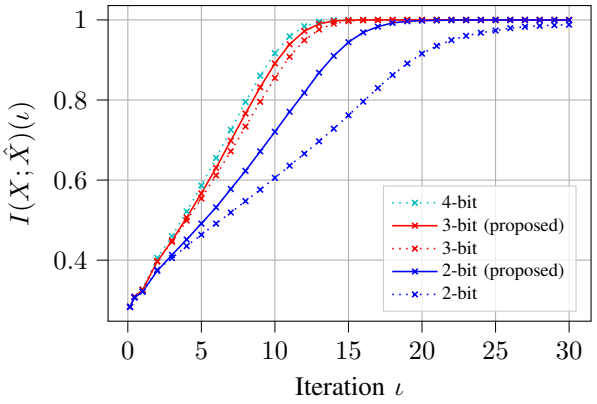


Fig. 5: Evolution of mutual information with code rate $1/3$. All results obtained with same design E_b/N_0=1.0 dB.

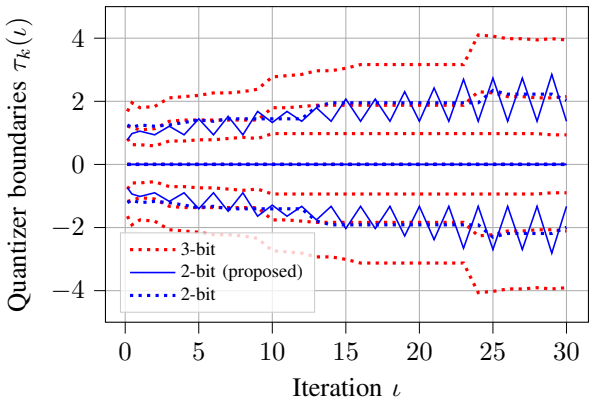


Fig. 6: Evolution of boundary placement each with code rate $1/3$ and individual design $E_b/N_0 = .55, .73$ and 1.0 dB.

C. Reduced Complexity with Merged Reconstruction Message

In the previous sections the side information s^c is equal to t^c from the previous iteration and is discarded after it has been used. Instead of discarding s^c, this section considers a side information $s^\Psi \in \{\pm 1,..,\pm 2^{w_\Psi - 1}\}$ which is merged with t^c into a compressed message $t^\Upsilon = \Upsilon(s^\Psi, t^c) \in \{\pm 1,..,\pm 2^{w_\Upsilon - 1}\}$ obtained through a compression table Υ. The table is designed using the IB method with a measured joint distribution $p(x, s^\Psi, t^c)$. The reconstruction in (5) now uses $\bar{L}(t^\Upsilon|x)$ instead of $\bar{L}(s^\Psi, t^c|x)$. The side information can propagate over many iterations as $s^\psi = \Psi(\Upsilon(\underline{s}^\psi, \underline{t}^c))$ where Ψ is another compression for reducing the table size of Υ.

The compression reduces the memory demand from $|\mathcal{N}|(w_\Psi + w_c)$ to $|\mathcal{N}|w_\Upsilon$ bits in a row-layered decoder structure depicted in Fig. 8. The row-layered schedule sequentially updates orthogonal sets of rows of $\mathbf{H}$ [11]. One layer update consists of partial VN and a full CN update (see section II-B). A partial VN update computes the VN-to-CN message as $t^v = Q(L^v)$ with $L^v = \hat{\underline{L}} - \underline{L}^c$. The reconstruction $\underline{L}^c = \underline{\phi}(\underline{t}^\Upsilon)$ translates $\underline{t}^\Upsilon$ from the previous iteration to a w'-bit integer representations of $\bar{L}(\underline{t}^\Upsilon|x)$. Significant loss is avoided if $w' \geq 7$ bits [11]. A permutation performs a cyclic shift of Z parallel messages. A full CN update computes CN-to-VN messages for all connected VNs according to (3). After an inverse permutation, a CN-to-VN message t^v is merged with the side information s^Ψ. Finally, $\hat{L} = L^v + \phi(t^\Upsilon)$.

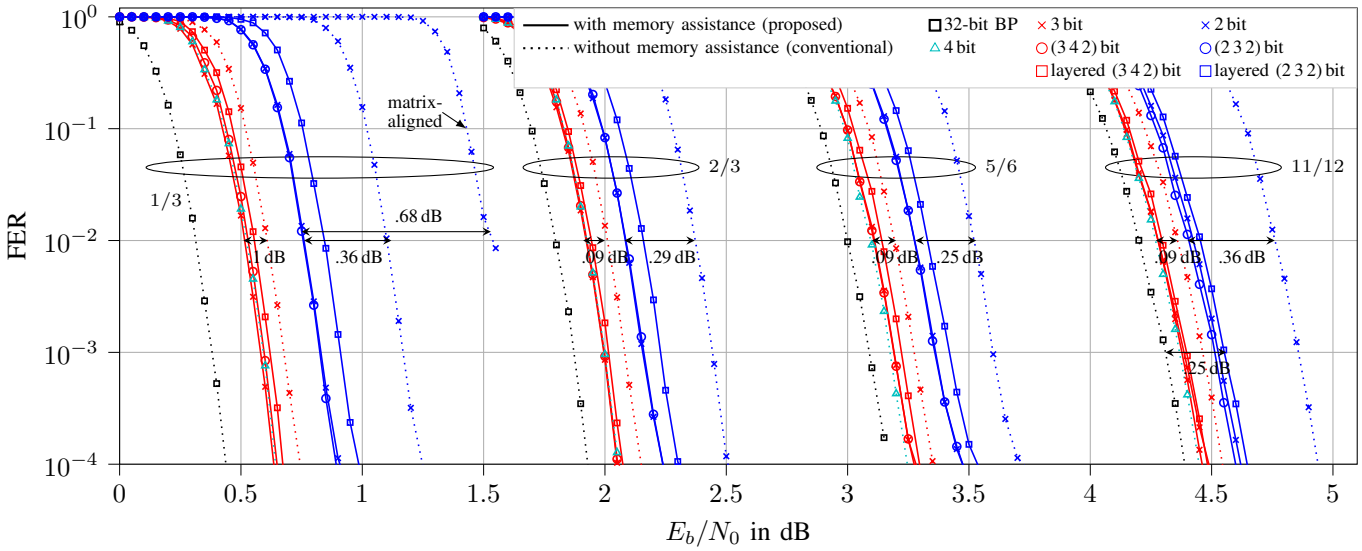


Fig. 7: Performance for rates $1/3$, $2/3$, $5/6$ and $11/12$ with and without memory assistance.

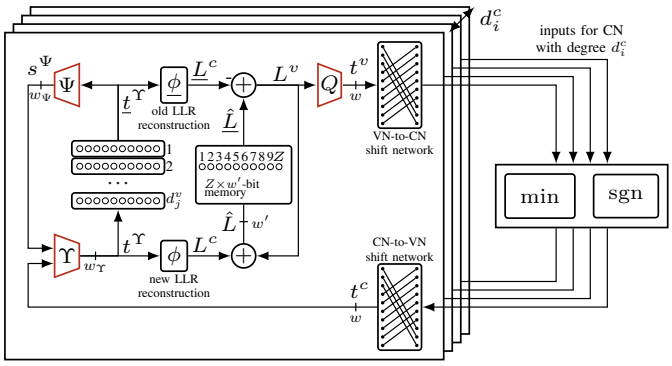


Fig. 8: Efficient structure of a row-layered decoder with memory-assistance and merged reconstruction messages t^{Υ}.

D. Frame Error Rate Performance

The decoding performance with different resolutions, code rates and schedules is depicted in Fig. 7. If not indicated otherwise all quantized decoders apply CN-aware quantization with row-alignment as in [11]. For rate $1/3$, the proposed w=2-bit (w=3) memory-assisted decoder improves performance by 0.36 dB (0.1 dB). Relative to a decoder with matrix alignment, no CN-aware quantization and no memory-assistance the performance gain is 0.68 dB. The w=3-bit memory-assisted decoder is able to slightly outperform a w=4-bit decoder. The labels (3 4 2) and (2 3 2) specify the bit widths ($w\, w_{\Upsilon}\, w_{\Psi}$) of the decoders with merged reconstruction message (see section V-C), respectively. They reduce the memory demand from 6 to 4 and 4 to 3 bits without losing performance compared to the non-merged decoders. The row-layered schedule with 15 iterations can achieve almost similar performance as the flooding schedule with 30 iterations. The improvements also translate to other code rates $2/3$ to $11/12$. Remarkably, under high rate $11/12$, the (2 3 2)-bit decoder operates within 0.25 dB compared to a belief propagation decoder with accurate box-plus operation and 32-bit LLR messages [3].

VI. Conclusions

This letter proposed coarsely quantized decoding which is assisted through messages retained in memory. We extended an existing IB algorithm for the design of threshold quantization which is aware of side information provided by the memory. Further, we proposed a new structure which merges the side information with a newly generated CN message to reduce the memory overhead. In summary, 2-bit (layered) decoding is improved up to 0.36 dB by increasing the memory from 2 to 3 bits, but preserving high speed 2-bit operations for quantization, routing network and the CN update.

References

[1] 3GPP, "5G NR: Multiplexing and Channel Coding, TS 38.212," 2018.
[2] R. Gallager, "Low-Density Parity-Check Codes," *IRE Transactions on Information Theory*, vol. 8, no. 1, pp. 21–28, 1962.
[3] J. Chen, A. Dholakia *et al.*, "Reduced-complexity decoding of LDPC codes," *IEEE Trans. on Commu.*, vol. 53, no. 8, pp. 1288–1299, 2005.
[4] P. Kang, K. Cai *et al.*, "Generalized Mutual Information-Maximizing Quantized Decoding of LDPC Codes With Layered Scheduling," *IEEE Trans. on Vehicular Technology*, vol. 71, no. 7, pp. 7258–7273, 2022.
[5] L. Wang, C. Terrill *et al.*, "Reconstruction-Computation-Quantization (RCQ): A Paradigm for Low Bit Width LDPC Decoding," *IEEE Trans. on Communications*, vol. 70, no. 4, pp. 2213–2226, 2022.
[6] M. Geiselhart, A. Elkelesh *et al.*, "Learning quantization in LDPC decoders," in *2022 IEEE Globecom Workshops*, 2022, pp. 467–472.
[7] Y. Ren, H. Harb *et al.*, "A Generalized Adjusted Min-Sum Decoder for 5G LDPC Codes: Algorithm and Implementation," 2024.
[8] J. Lewandowsky and G. Bauch, "Information-Optimum LDPC Decoders Based on the Information Bottleneck Method," *IEEE Access*, vol. 6, pp. 4054–4071, 2018.
[9] M. Stark, L. Wang *et al.*, "Decoding Rate-Compatible 5G-LDPC Codes With Coarse Quantization Using the Information Bottleneck Method," *IEEE Open Journal of the Comm. Society*, vol. 1, pp. 646–660, 2020.
[10] P. Mohr and G. Bauch, "A Variable Node Design with Check Node Aware Quantization Leveraging 2-Bit LDPC Decoding," in *GLOBECOM 2022 - 2022 IEEE Global Communications Conf.*, 2022, pp. 3484–3489.
[11] ——, "Region-Specific Coarse Quantization with Check Node Awareness in 5G-LDPC Decoding," *arXiv:2406.14233*, 2024.
[12] G. Chechik and N. Tishby, "Extracting Relevant Structures with Side Information," in *Advances in Neural Inf. Proc. Sys.* MIT Press, 2002.
[13] S. Steiner and V. Kuehn, "Distributed compression using the information bottleneck principle," in *ICC 2021 - IEEE International Conference on Communications*, 2021, pp. 1–6.
[14] B. M. Kurkoski and H. Yagi, "Quantization of Binary-Input Discrete Memoryless Channels," *IEEE Transactions on Information Theory*, vol. 60, no. 8, pp. 4544–4552, Aug. 2014.