Mixed Block Markov Superposition Transmission Codes

Philipp Mohr, Jasper Brüggmann, Viet Hoang Le, and Gerhard Bauch, *Fellow, IEEE*

***Abstract*—Block Markov superposition transmission (BMST) codes provide a flexible framework for constructing codes with near-capacity performance and low-complexity sliding-window decoding. However, existing BMST variants show contrasting performance limitations: recursive BMST (rBMST) codes suffer from error propagation but avoid high error floors, whereas non-recursive BMST codes exhibit the opposite behavior. Motivated by these complementary characteristics, we combine recursive and non-recursive components through parallel and serial concatenation, yielding mixed BMST (mBMST) codes. The proposed framework subsumes existing BMST variants and enables new BMST structures. Simulations show that these structures improve FER and BER performance with lower memory requirements than rBMST.**

***Index Terms*—Normal graph, BMST, Feedforward, Feedback, Recursive, Non-recursive, 6G.**

I. Introduction

FORWARD error-correcting (FEC) codes are a key enabler of modern communication systems, driven by increasingly stringent requirements on throughput, latency, reliability, and energy efficiency. This evolution is reflected in mobile communication standards, which have progressed from feedforward convolutional codes in 2G systems to turbo codes employing recursive components in 3G/4G, and to low-density parity-check (LDPC) codes in 5G New Radio (NR).

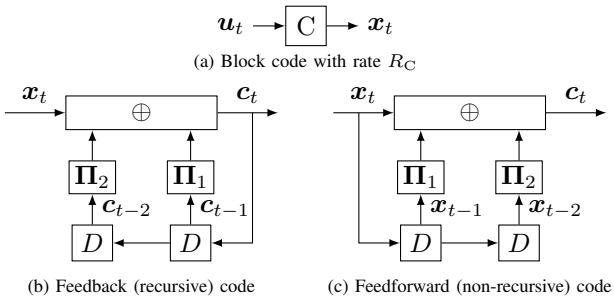
A promising direction for further performance improvement is spatial coupling, in which code blocks are connected through extended code constraints [1], [2], [3]. Several classes of spatially coupled codes exhibit the threshold saturation effect, whereby the belief-propagation (BP) decoding threshold approaches the maximum-a-posteriori (MAP) threshold of the underlying block code [2]. BP decoding of spatially coupled codes can be efficiently implemented using a sliding-window algorithm [1].

Block Markov superposition transmission (BMST) codes constitute an important class of spatially coupled codes [4], [5], [6], [7]. In BMST, multiple *basic* codewords are superimposed across successive transmission blocks using a rate-1 convolutional component. BMST codes are attractive due to their flexible construction, multi-rate capability, and simple encoding. Furthermore, the regular graph structure provides high decoding locality, since each decoding step involves only a few neighboring blocks.

In this letter, we show that BMST codes employing recursive components (rBMST) suffer from error propagation but achieve low error floors, whereas non-recursive BMST (nBMST) codes exhibit the opposite behavior. To combine the strengths of both approaches, we propose a parallel concatenation of recursive and non-recursive components, leading to performance improvements of up to $0.14\,\mathrm{dB}$. The proposed alternating decoding schedule also offers potential implementation advantages. Furthermore, we generalize parallel and serial concatenation principles within the BMST framework, leading to the class of *mixed BMST (mBMST) codes*. The proposed framework subsumes several existing constructions, including nBMST [4], rBMST [5], DrBMST [6], and BiBMST [7].

This work is supported by Huawei Technologies Sweden AB. (Corresponding author: Philipp Mohr.)

The authors are with the Institute of Communications, Hamburg University of Technology, 21073 Hamburg, Germany (e-mail: philipp.mohr@tuhh.de; jasper.brueggmann@tuhh.de; viet.le@tuhh.de; bauch@tuhh.de).



(a) Block code with rate R_C

(b) Feedback (recursive) code

(c) Feedforward (non-recursive) code

Fig. 1. Constituent codes for BMST (memory $m = 2$ in (b) and (c)).

II. Block Markov Superposition Transmission

A. Encoding

The encoding principle of BMST is depicted in Fig. 1. A sequence of L data blocks $\boldsymbol{u}_t \in \mathbb{F}_2^K$ is encoded, where t denotes a time index and the block length is K. Each block $\boldsymbol{u}_t$ is mapped using a basic code $\mathrm{C}[N, K]$ to a codeword $\boldsymbol{x}_t \in \mathbb{F}_2^N$, as depicted in Fig. 1a. The individual codewords $\boldsymbol{x}_t$ are superimposed in time using a rate-1 block-convolutional encoder with memory m to form the transmitted codeword $\boldsymbol{c}_t \in \mathbb{F}_2^N$. Two fundamental convolutional encoder variants exist with a feedback (recursive) structure, shown in Fig. 1b,

$$\boldsymbol{c}_t = \mathrm{C}_m^{\mathrm{r}}(\boldsymbol{x}_t) \triangleq \boldsymbol{x}_t \oplus \boldsymbol{\Pi}_1 \boldsymbol{c}_{t-1} \oplus \ldots \oplus \boldsymbol{\Pi}_m \boldsymbol{c}_{t-m} \tag{1}$$

or a feedforward (non-recursive) structure, shown in Fig. 1c,

$$\boldsymbol{c}_t = \mathrm{C}_m^{\mathrm{n}}(\boldsymbol{x}_t) \triangleq \boldsymbol{x}_t \oplus \boldsymbol{\Pi}_1 \boldsymbol{x}_{t-1} \oplus \ldots \oplus \boldsymbol{\Pi}_m \boldsymbol{x}_{t-m} \tag{2}$$

where random permutation matrices $\boldsymbol{\Pi}_i$ are used. The overall number of data blocks is L, followed by a termination sequence $\boldsymbol{u}_t = \boldsymbol{0}$ for $t = L{+}1, \ldots, L{+}T$. The rate of the BMST system is $R = R_\mathrm{C} \cdot \frac{L}{L+T}$ with basic code rate $R_\mathrm{C} = K/N$.

The BMST code constraint can be represented using the normal graph framework [8], as shown in Fig. 2 for the recursive encoder in (1) and the non-recursive encoder in (2), respectively. Each edge of the graph represents a block of variables, e.g., $\boldsymbol{c}_t$ or $\boldsymbol{u}_t$. The graph consists of repetition (REP) nodes $\boxed{=}$, single parity check (SPC) nodes $\boxed{+}$, and basic code nodes $\boxed{\mathrm{C}}$. The permutations $\boldsymbol{\Pi}_i$ are integrated into the edges of the graph for brevity.

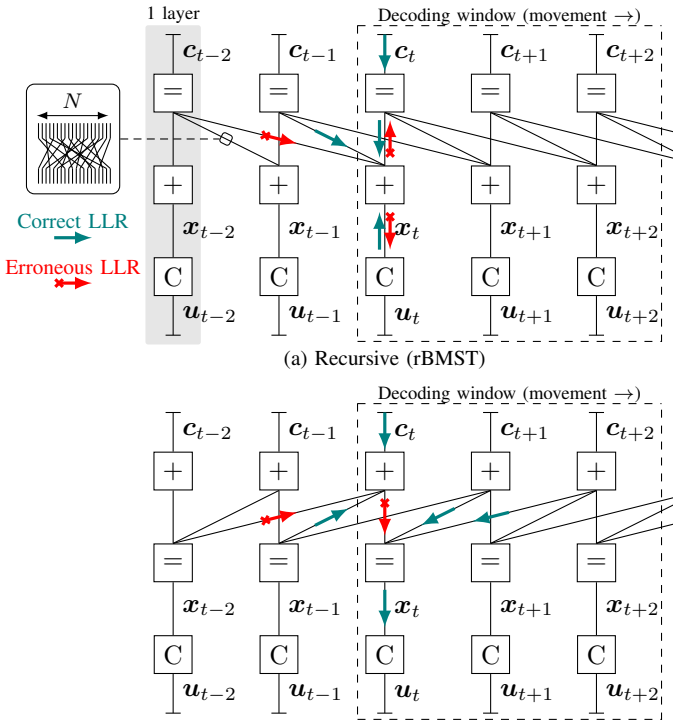


Fig. 2. Error propagation in recursive and non-recursive BMST graphs.

B. Decoding

Each block $\boldsymbol{c}_t$ is modulated using binary phase-shift keying, so that $1-2\boldsymbol{c}_t$ is transmitted and the channel output is $\boldsymbol{y}_t = 1-2\boldsymbol{c}_t+\boldsymbol{n}_t$. We consider additive white Gaussian noise $\boldsymbol{n}_t \in \mathbb{R}^N$ with variance $N_0/2$. The decoder observes the LLRs $\boldsymbol{\ell}_t^{\text{ch}} = 4\boldsymbol{y}_t/N_0$. A sliding window restricts processing to W consecutive layers. In Fig. 2a, one layer includes the nodes $\boxed{=}$, $\boxed{+}$, and $\boxed{\text{C}}$. At each window position, the decoder performs I_{W} iterations. One iteration consists of a forward sweep and a backward sweep over all W layers in the decoding window. At each visited layer, the node updates are applied sequentially according to the following schedule:

- $\boxed{=} \to \boxed{+} \to \boxed{\text{C}} \to \boxed{+} \to \boxed{=}$ (recursive code)
- $\boxed{+} \to \boxed{=} \to \boxed{\text{C}} \to \boxed{=} \to \boxed{+}$ (non-recursive code)

In this work the basic code node $\boxed{\text{C}}$ is a rate-1/2 REP node $\boxed{=}$. Each node update in the schedule computes LLR messages $\ell_n^{\beta\to\gamma}$ from node β to node γ using extrinsic input messages $\ell_n^{\alpha\to\beta}$ for $n=1,\dots,N$. The following update rules are used

$$\ell_n^{\beta\to\gamma} = \begin{cases} \sum\limits_{\alpha\in\mathcal{N}(\beta)\setminus\gamma} \ell_n^{\alpha\to\beta} & \text{if } \beta \text{ is a REP node } \boxed{=} \\ \boxplus\limits_{\alpha\in\mathcal{N}(\beta)\setminus\gamma} \ell_n^{\alpha\to\beta} & \text{if } \beta \text{ is an SPC node } \boxed{+} \end{cases} \tag{3}$$

where $\mathcal{N}(\beta)$ denotes the set of all nodes connected to β. The boxplus operation is defined as

$$\ell_1 \boxplus \ell_2 \triangleq \operatorname{sgn}(\ell_1)\operatorname{sgn}(\ell_2)(\min(|\ell_1|,|\ell_2|) - \mu(|\ell_1|,|\ell_2|)) \tag{4}$$

$$\mu(|\ell_1|,|\ell_2|) \triangleq \log\frac{1+\exp(-||\ell_1|-|\ell_2||)}{1+\exp(-(|\ell_1|+|\ell_2|))} \tag{5}$$

where μ is a non-negative correction function.

III. Properties of Recursive/Non-Recursive BMST

A. Error Propagation

For rBMST, the feedback path in Fig. 1b causes any bit error in $\boldsymbol{x}_t$ to propagate indefinitely through the recursive structure. In contrast, the nBMST encoder shown in Fig. 1c limits the effect of a bit error in $\boldsymbol{x}_t$ to m subsequent blocks. This behavior indicates that the recursive structure is more prone to error propagation.

From the decoder perspective, we need to consider processing of LLR messages. In Fig. 2, an edge case is shown where the decoding window covers the layers t to $t+2$. Consider one of the N LLRs originating from layer $t-2$ to be erroneous denoted by ↛. This LLR will not be corrected, as layer $t-2$ is outside the window.

In the rBMST graph (cf. Fig. 2a) at t, the SPC node $\boxed{+}$ propagates the erroneous LLR ↛ if the remaining input LLRs → are correct. Thus, the basic code node $\boxed{\text{C}}$ and REP node $\boxed{=}$ receive erroneous LLRs ↛, which may propagate into later layers.

In the non-recursive graph (cf. Fig. 2b), an erroneous LLR ↛ from layer $t-2$ can be corrected by the REP node $\boxed{=}$ update rule in (3) before being propagated to the basic code node $\boxed{\text{C}}$ or SPC nodes $\boxed{+}$.

Thus, in the non-recursive structure, an erroneous LLR from a previous layer can be corrected, whereas in the recursive structure it propagates to multiple connected nodes.

B. Error Correction Capability

Although less prone to error propagation, the non-recursive structure has significant limitations in terms of error correction capability. Let $\mathtt{s}$, $\mathtt{r}$, and $\mathtt{c}$ denote SPC $\boxed{+}$, REP $\boxed{=}$, and basic code $\boxed{\text{C}}$ nodes, respectively. The channel messages $\ell^{\mathtt{ch}\to\mathtt{s}}$ cannot improve during the iterations. Hence, the LLR magnitudes $|\ell^{\mathtt{s}\to\mathtt{r}}|$ are bounded by the channel LLR magnitudes due to the boxplus operation in (4). In turn, the LLR magnitudes $|\ell^{\mathtt{r}\to\mathtt{c}}|$ are bounded by the sum of the corresponding channel LLR magnitudes due to the REP update rule in (3):

$$\max(|\ell^{\mathtt{r}\to\mathtt{c}}|) = \sum_{\mathtt{s}\in\mathcal{N}(\mathtt{r})\setminus\mathtt{c}} |\ell^{\mathtt{ch}\to\mathtt{s}}| \tag{6}$$

In the case of a rate-$1/2$ REP basic code, two LLR messages $\ell^{\mathtt{r}_1\to\mathtt{c}}$ and $\ell^{\mathtt{r}_2\to\mathtt{c}}$ are involved in the information bit decision:

$$\hat{u} = \begin{cases} 0 & \text{for } \ell^{\mathtt{r}_1\to\mathtt{c}} + \ell^{\mathtt{r}_2\to\mathtt{c}} > 0 \\ 1 & \text{otherwise} \end{cases} \tag{7}$$

An erroneous LLR with magnitude $|\ell^{\mathtt{r}_2\to\mathtt{c}}| > \max(|\ell^{\mathtt{r}_1\to\mathtt{c}}|)$ cannot be corrected by a correct LLR $\ell^{\mathtt{r}_1\to\mathtt{c}}$.

The recursive structure, on the other hand, does not exhibit the limitation in (6). The LLRs exchanged between REP and SPC nodes can grow across iterations without being bounded by $m+1$ channel messages.

IV. Mixed BMST via Parallel Concatenation

The following proposes a structure that aims at combining the strengths of the recursive and non-recursive structures discussed in section III. The scheme is a parallel concatenation

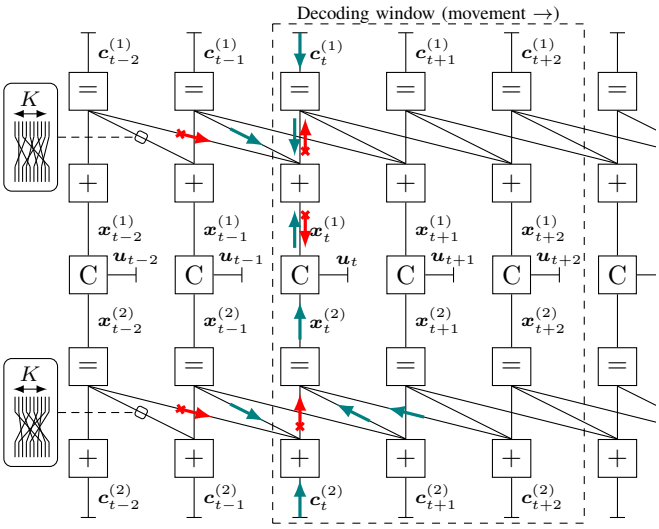


Fig. 3. Error propagation in a mixed BMST graph. The non-recursive branch 2 is less prone to propagating errors into the recursive branch 1.

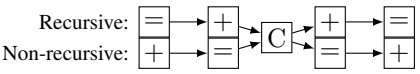


Fig. 4. Schedule for updating a layer under mixed BMST decoding.

of the convolutional rate-1 codes. After encoding $\boldsymbol{u}_t$ to $\boldsymbol{x}_t$ with the basic code, a branch operation b divides $\boldsymbol{x}_t$ into B segments $\boldsymbol{x}_t^{(b)}$.

In this letter, we restrict attention to the exemplary case $B = 2$. Since a rate-$1/2$ REP basic code is employed, the segments can be defined as $\boldsymbol{x}_t^{(1)} = \boldsymbol{u}_t$ and $\boldsymbol{x}_t^{(2)} = \boldsymbol{u}_t$. The first branch is encoded using a recursive code, whereas the second branch is encoded using a non-recursive code:

$$\boldsymbol{c}_t = \begin{bmatrix} \boldsymbol{c}_t^{(1)} \\ \boldsymbol{c}_t^{(2)} \end{bmatrix} = \begin{bmatrix} \mathrm{C}_{m_1}^{\mathrm{r}}(\boldsymbol{x}_t^{(1)}) \\ \mathrm{C}_{m_2}^{\mathrm{n}}(\boldsymbol{x}_t^{(2)}) \end{bmatrix} \tag{8}$$

Fig. 3 shows the corresponding normal graph. Fig. 4 describes a schedule for a layer update, where the recursive schedule can run concurrently with the non-recursive schedule. This structure may improve implementation efficiency: Updating a single LLR requires either a REP or an SPC computation module, as defined in (3). In the mixed structure, each layer comprises two branches with K parallel edges each. A fully parallel implementation therefore requires K REP modules and K SPC modules, which can operate concurrently. By contrast, the recursive structure in Fig. 2a has a single branch per layer with N parallel edges. Consequently, a fully parallel layer update requires N REP modules and N SPC modules, which are executed sequentially. Assuming identical delay for all modules, this implies that the mixed structure reduces the required implementation area by $K/N = 1/2$ while achieving the same throughput. An additional advantage is that LLRs in the non-recursive structure may be represented with fewer bits under quantization, since their dynamic range is limited by the channel messages (cf. section III-B). Moreover, permutations of size K are less costly than permutations of size N.

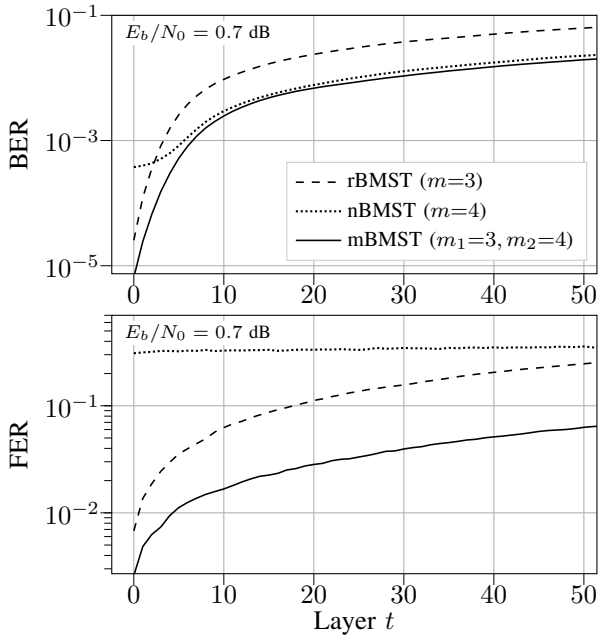


Fig. 5. BER/FER per layer (info block length K=1000, code rate $R_{\mathrm{C}} = 1/2$, coupling length L=500, window size W=11, termination length T=W−1, iteration count I_{W}=10).

A. Numerical Analysis

1) Error Propagation Behavior: Fig. 5 compares the bit error rate (BER) and frame error rate (FER) for recursive (rBMST), non-recursive (nBMST), and mixed (mBMST) configurations. A frame error is counted for layer t if at least one bit error occurs in the estimate of the information block $\boldsymbol{u}_t$ after the sliding window has completed all iterations. For all configurations, the BER and FER are lowest at the beginning, since the initial symbols stored in the encoder memory are known. As the layer index t increases, both metrics degrade due to error propagation from previous layers.

For nBMST, the BER is lower than for rBMST, while the FER is higher. This indicates that frame errors typically contain only a small number of bit errors. A contributing factor is that sequences of low-magnitude channel LLRs can prevent error correction, as discussed in section III-B. In contrast, rBMST exhibits a lower FER but a higher BER, implying that frame errors tend to contain many bit errors. This behavior can be attributed to recursive error propagation, where a single bit error entering the next layer can trigger additional errors, as discussed in section III-A. The proposed mBMST configuration balances these effects and achieves the lowest BER and FER across all layers.

2) Iterative Behavior: This section studies how the quality of LLR messages evolves across decoding iterations. From the perspective of a layer t, the sliding window gradually moves over it. At each window position, the layer undergoes two updates in each of the I_{W} iterations. Exceptions occur when the layer t is the leftmost or rightmost layer in the sliding window, where only a single layer update is applied per iteration. For a window size W, a layer therefore experiences $2I_{\mathrm{W}}(W-1)$ updates.

Let $\boldsymbol{\ell}_t^{(b)}$ denote the LLR message vector generated by

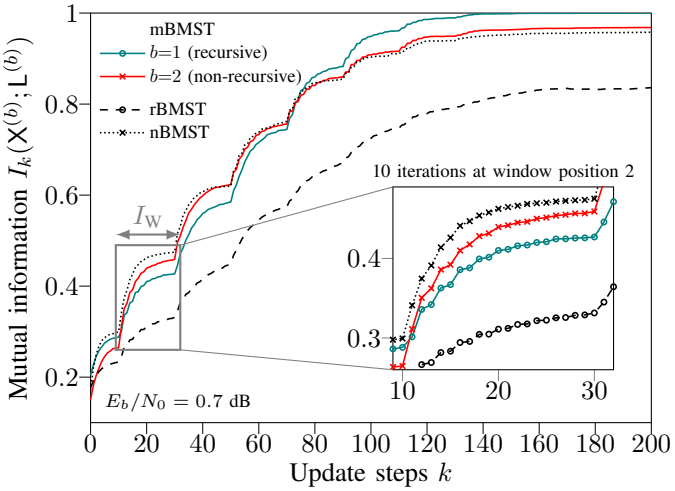
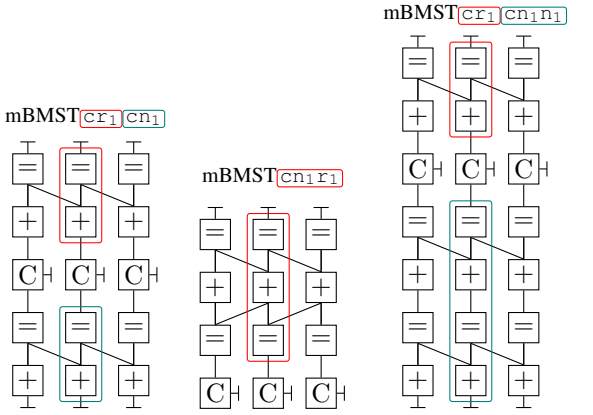


Fig. 6. Evolution of mutual information for layer $t = 25$ from Fig. 5.



(a) Mixed parallel (b) Mixed serial (c) Mixed parallel & serial

Fig. 7. Mixed configurations with recursive and non-recursive components.

branch b which is input to the basic code node C. The LLRs in $\boldsymbol{\ell}_t^{(b)}$ and the corresponding code bits in $\boldsymbol{x}_t^{(b)}$ can be regarded as realizations of random variables $\mathsf{L}^{(b)}$ and $\mathsf{X}^{(b)}$, respectively, whose statistics are estimated empirically from simulation data. Fig. 6 shows the mutual information $I_k(\mathsf{X}^{(b)}, \mathsf{L}^{(b)})$ for the mBMST, rBMST, and nBMST decoders from Fig. 5, evaluated for layer $t = 25$. The update steps are indexed by $k = 1, \ldots, 2I_W(W-1)$. LLRs $\boldsymbol{\ell}_t^{(b)}$ are called *informative* if $I_k(\mathsf{X}^{(b)}, \mathsf{L}^{(b)}) > 0$.

We first analyze the mutual information evolution for the two branches of the mBMST configuration. As specified in (8), the recursive component corresponds to branch 1 and the non-recursive component to branch 2. For steps 1 to 10, layer 25 is the right-most layer in the sliding window, similarly as the layer $t+2$ in Fig. 3. In this range, the mutual information of branch 2 is lower than that of branch 1. This is because only the first of the $m+1$ connected SPC nodes to the REP node in branch 2 provides informative LLRs. These LLRs are bounded by the channel LLRs entering the SPC node. In contrast, the SPC node in branch 1 is connected to $m+1$ REP nodes, each contributing informative LLRs. Once the window advances by one position, the REP node in branch 2 receives two informative LLRs. As a result, $I_k(\mathsf{X}^{(2)}, \mathsf{L}^{(2)})$ surpasses $I_k(\mathsf{X}^{(1)}, \mathsf{L}^{(1)})$ at step 10. After step 80, a second crossing point occurs where $I_k(\mathsf{X}^{(1)}, \mathsf{L}^{(1)})$ becomes dominant, since the LLR magnitudes from branch 2 are bounded as discussed in section III-B. Ultimately, only $I_k(\mathsf{X}^{(1)}, \mathsf{L}^{(1)})$ converges to 1.0, whereas $I_k(\mathsf{X}^{(2)}, \mathsf{L}^{(2)})$ saturates at approximately 0.95 for the chosen E_b/N_0.

Looking at the conventional configurations, the rBMST decoder does not converge, while the nBMST decoder exhibits high mutual information initially but also saturates, similar to branch 2 in the mBMST configuration.

V. Parallel and Serial Concatenation

A. Generalization

In addition to parallel concatenation with branches $b \in \{1, \ldots, B\}$, serial concatenation with stages $s \in \{1, \ldots, S_b\}$ can be applied within each branch. The output of stage $s+1$ in branch b is given by

$$\boldsymbol{x}_t^{(b,s+1)} \triangleq \mathrm{C}_{b,s}(\boldsymbol{x}_t^{(b,s)}), \tag{9}$$

where the initialization at stage 1 is $\boldsymbol{x}_t^{(b,0)} = \boldsymbol{x}_t^{(b)}$, and the output of stage S_b is denoted by $\boldsymbol{c}_t^{(b)} = \boldsymbol{x}_t^{(b,S_b)}$. Each component code $\mathrm{C}_{b,s}$ is characterized by a memory length $m_{b,s}$ and operates either in recursive or non-recursive mode:

$$\mathrm{C}_{b,s}(\boldsymbol{x}_t^{(b,s)}) = \begin{cases} \mathrm{C}^{\mathrm{r}}_{m_{b,s}}(\boldsymbol{x}_t^{(b,s)}) & \text{recursive,} \\ \mathrm{C}^{\mathrm{n}}_{m_{b,s}}(\boldsymbol{x}_t^{(b,s)}) & \text{non-recursive.} \end{cases} \tag{10}$$

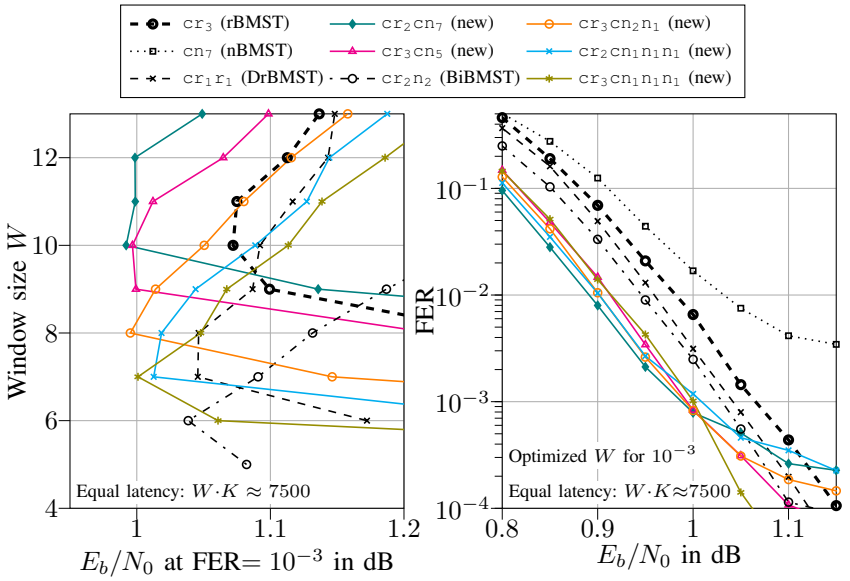
The overall codeword is then obtained as

$$\boldsymbol{c}_t = \begin{bmatrix} \boldsymbol{c}_t^{(1)} \\ \vdots \\ \boldsymbol{c}_t^{(B)} \end{bmatrix} = \begin{bmatrix} \mathrm{C}_{1,S_1}(\ldots(\mathrm{C}_{1,1}(\boldsymbol{x}_t^{(1)}))) \\ \vdots \\ \mathrm{C}_{B,S_B}(\ldots(\mathrm{C}_{B,1}(\boldsymbol{x}_t^{(B)}))) \end{bmatrix}. \tag{11}$$

Fig. 7 depicts the graphs corresponding to several possible configurations defined by (11) with $B \leq 2$. The naming scheme, such as $\mathtt{cr}_1\mathtt{cn}_1\mathtt{n}_1$, provides a compact representation of the code structure. The symbol c denotes the beginning of a branch starting from a basic code. A serial concatenation of recursive and non-recursive components within a branch is specified by a sequence of r and n symbols. An additional c indicates the presence of a second branch corresponding to parallel concatenation. Optional subscripts specify the corresponding memories $m_{b,s}$. Existing schemes such as BMST [4], rBMST [5], DrBMST [6], and BiBMST [7] are denoted by cn, cr, crr and crn, respectively. New configurations use parallel concatenation (e.g. crcn), serial concatenation (e.g. cnr) or parallel & serial concatenation (e.g. crcnn).

B. Performance Analysis

For fair comparisons, the BER and FER performance of different configurations is compared at constant decoding latency $W \cdot K$. The ratio W/K can be optimized for each configuration. In addition the coupling length L is chosen such that the rate satisfies $R = R_{\mathrm{C}}(L/L+T) = 0.49$ with termination length $T = W - 1$.



Category	Configuration	E_b/N_0@ BER 10^{-4}	E_b/N_0@ FER 10^{-3}	Optimized W for BER	Optimized W for FER	#BE per FE @ FER 10^{-3}
rBMST	$\mathtt{cr}_3$	1.10 dB	1.07 dB	10	10	230
nBMST	$\mathtt{cn}_7$	−.11	+.52	8	13	1.0
Serial	$\mathtt{cr}_1\mathtt{r}_1$	−.04	−.03	7	7	170
	$\mathtt{cr}_2\mathtt{r}_2$	−.00	−.03	6	6	428
	$\mathtt{cr}_1\mathtt{r}_1\mathtt{r}_1$	−.01	−.03	5	5	400
Mixed parallel	$\mathtt{cr}_2\mathtt{cn}_7$	−.13	−.08	9	10	136
	$\mathtt{cr}_3\mathtt{cn}_5$	−.11	−.08	8	10	287
Mixed serial	$\mathtt{cr}_2\mathtt{n}_2$	−.05	−.03	5	6	479
	$\mathtt{cn}_2\mathtt{r}_2$	−.01	−.03	5	5	416
Mixed parallel & serial	$\mathtt{cr}_1\mathtt{r}_1\mathtt{cn}_5$	−.12	−.02	7	11	156
	$\mathtt{cr}_1\mathtt{r}_1\mathtt{cn}_2\mathtt{n}_2$	−.10	−.05	5	7	297
	$\mathtt{cr}_2\mathtt{cn}_1\mathtt{n}_1$	−.01	+1.75	11	12	1.4
	$\mathtt{cr}_2\mathtt{cn}_2\mathtt{n}_2$	−.10	−.08	6	7	175
	$\mathtt{cr}_3\mathtt{cn}_1\mathtt{n}_1$	−.11	+.65	9	12	1.4
	$\mathtt{cr}_3\mathtt{cn}_2\mathtt{n}_1$	−.09	−.08	7	8	224
	$\mathtt{cr}_3\mathtt{cn}_2\mathtt{n}_2$	−.08	−.07	6	7	318
	$\mathtt{cr}_4\mathtt{cn}_1\mathtt{n}_1$	−.11	−.03	8	11	130
	$\mathtt{cr}_2\mathtt{cn}_1\mathtt{n}_1\mathtt{n}_1$	−.11	−.06	6	7	140
	$\mathtt{cr}_3\mathtt{cn}_1\mathtt{n}_1\mathtt{n}_1$	−.10	−.07	6	7	312
	$\mathtt{cr}_3\mathtt{cn}_3\mathtt{n}_2\mathtt{n}_1$	−.06	−.04	5	6	339
	$\mathtt{cr}_2\mathtt{n}_2\mathtt{cn}_4$	−.10	+.31	6	10	1.2
	$\mathtt{cr}_3\mathtt{n}_5\mathtt{cn}_5$	−.06	−.04	6	7	359
	$\mathtt{cr}_2\mathtt{n}_2\mathtt{cn}_2\mathtt{r}_2$	−.00	−.03	5	6	475

Fig. 8. BER/FER of mixed BMST configurations. The decoding latency $W \cdot K$ is kept constant. FER is evaluated with frame length of 1000.

In Fig. 8 we provide and overview of the performance with many configurations. A strong dependence on the window size W can be noticed. A serial concatenation tends to favor smaller W whereas the parallel concatenation benefits from larger W. Furthermore, the table in Fig. 8 shows, that optimizing for BER performance leads to smaller W than optimizing for FER performance.

The baseline configuration, $\mathtt{cr}_3$ (rBMST), achieves an FER of 10^{-3} at $E_b/N_0 = 1.07\,\mathrm{dB}$ and a BER of 10^{-4} at $E_b/N_0 = 1.10\,\mathrm{dB}$. The $\mathtt{cn}_7$ (nBMST) decoder can achieve better BER performance, but requires substantially more memory and still exhibits a high FER error floor. At FER of 10^{-3}, each erroneous frame contains a single bit error (see the last column of the table), resulting from the limited error correction capability discussed in section III-B.

A significantly improved memory-performance trade-off is achieved using mixed configurations. For example, $\mathtt{cr}_2\mathtt{cn}_7$ requires only a memory of 2 in the first branch and improves the BER and FER performance by $0.13\,\mathrm{dB}$ and $0.08\,\mathrm{dB}$, respectively. The memory requirements can be further reduced through serial concatenation in the non-recursive branch. For example, compared to $\mathtt{cr}_2\mathtt{cn}_7$, $\mathtt{cr}_2\mathtt{cn}_1\mathtt{n}_1\mathtt{n}_1$ reduces non-recursive memory from 7 to 3, with comparable performance. Increasing memory in the recursive branch, as in $\mathtt{cr}_3\mathtt{cn}_1\mathtt{n}_1\mathtt{n}_1$, can improve the error floor performance. Other promising candidates use a decreasing memory in the non-recursive branch, e.g., $\mathtt{cr}_3\mathtt{cn}_2\mathtt{n}_1$. However, the memory assignment must be well balanced with the recursive branch. For example, $\mathtt{cr}_3\mathtt{cn}_3\mathtt{n}_2\mathtt{n}_1$ performs significantly worse than $\mathtt{cr}_2\mathtt{cn}_1\mathtt{n}_1\mathtt{n}_1$.

Configurations available in the literature, such as $\mathtt{cr}_1\mathtt{r}_1$ (DrBMST) [6] and $\mathtt{cr}_2\mathtt{n}_2$ (BiBMST) [7], achieve FER gains of $0.03\,\mathrm{dB}$ over rBMST. In comparison, the proposed configurations achieve an FER improvement of $0.08\,\mathrm{dB}$, representing a noticeable gain over the state of the art.

VI. Conclusions

A new class of BMST variants, termed mixed BMST (mBMST) codes, combines recursive and non-recursive component codes through parallel and serial concatenation. Parallel mixing yields performance improvements of up to $0.13\,\mathrm{dB}$ compared to rBMST codes. Serial concatenation enables a reduction in encoding memory complexity while maintaining comparable performance. Open issues include the impact of the choice of the basic code, such as its rate and strength, as well as the behavior of systems with more than two branches. Also exploiting the mixed schedule in hardware implementations remains an important direction for future work.

References

[1] A. J. Felstrom and K. S. Zigangirov, "Time-varying periodic convolutional codes with low-density parity-check matrix," *IEEE Transactions on Information Theory*, vol. 45, no. 6, pp. 2181–2191, 1999.

[2] S. Kudekar, T. Richardson, and R. L. Urbanke, "Spatially coupled ensembles universally achieve capacity under belief propagation," *IEEE Transactions on Information Theory*, vol. 59, no. 12, pp. 7761–7813, 2013.

[3] Y. Ren, L. Zhang, Y. Shen, W. Song, E. Boutillon, A. Balatsoukas-Stimming, and A. Burg, "Edge-spreading Raptor-like LDPC codes for 6G wireless systems," *IEEE Transactions on Communications*, 2025.

[4] X. Ma, C. Liang, K. Huang, and Q. Zhuang, "Block Markov Superposition Transmission: Construction of Big Convolutional Codes From Short Codes," *IEEE Transactions on Information Theory*, vol. 61, no. 6, pp. 3150–3163, Jun. 2015.

[5] S. Zhao, X. Ma, Q. Huang, and B. Bai, "Recursive Block Markov Superposition Transmission of Short Codes: Construction, Analysis, and Applications," *IEEE Transactions on Communications*, vol. 66, no. 7, pp. 2784–2796, 2018.

[6] ——, "Doubly-Recursive Block Markov Superposition Transmission: A Low-Complexity and Flexible Coding Scheme," *IEEE Transactions on Communications*, vol. 68, Aug. 2020.

[7] G. Li, S. Zhao, H. Chen, and J. Wen, "Spatially Coupled Codes via Bidirectional Block Markov Superposition Transmission," *Tsinghua Science and Technology*, vol. 29, no. 3, pp. 656–670, 2024.

[8] G. Forney, "Codes on Graphs: Normal Realizations," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 520–548, Feb. 2001.