Low-Complexity Near-Optimum Symbol Detection Based on Neural Enhancement of Factor Graphs

Luca Schmid and Laurent Schmalen, *Senior Member, IEEE*

***Abstract*— We consider the application of the factor graph framework for symbol detection on linear inter-symbol interference channels. Based on the Ungerboeck observation model, a detection algorithm with appealing complexity properties can be derived. However, since the underlying factor graph contains cycles, the sum-product algorithm (SPA) yields a suboptimal algorithm. In this paper, we develop and evaluate efficient strategies to improve the performance of the factor graph-based symbol detection by means of neural enhancement. In particular, we consider neural belief propagation and generalizations of the factor nodes as an effective way to mitigate the effect of cycles within the factor graph. By applying a generic preprocessor to the channel output, we propose a simple technique to vary the underlying factor graph in every SPA iteration. Using this dynamic factor graph transition, we intend to preserve the extrinsic nature of the SPA messages which is otherwise impaired due to cycles. Simulation results show that the proposed methods can massively improve the detection performance, even approaching the maximum a posteriori performance for various transmission scenarios, while preserving a complexity which is linear in both the block length and the channel memory.**

***Index Terms*— Factor graphs, neural belief propagation, symbol detection, channels with memory, high-level parallelism.**

I. Introduction

THE well-known task of data transmission over a channel with linear inter-symbol interference (ISI) impaired by additive white Gaussian noise (AWGN) is considered in this paper. ISI is ubiquitous in many wireline and wireless communication systems where, e.g., multipath propagation is caused by reflections and refraction of the transmit signal in the wireless channel. Left uncompensated, ISI leads to a distortion of the signal and causes high error rates at the receiver [2, Chap. 9]. Detection algorithms are thus required at the receiver in order to recover the original transmit signal. Optimum detection with respect to the symbol error probability is based on maximum a posteriori (MAP) symbol detection. The BCJR algorithm [3] is an efficient MAP algorithm whose complexity is linear in the block length but exponential in the memory of the channel and the number of bits mapped to each constellation symbol. In many practical scenarios where channels have large memory or where high-order constellations are used, the BCJR algorithm becomes prohibitively complex. Therefore, the development of computationally efficient algorithms with near-optimum performance has become a major field of research. Classical low-complexity equalizers like linear transversal filters and algorithms based on decision-feedback equalization (DFE) yield acceptable performance for a wide range of communication channels with well behaved spectral characteristics, but perform poorly for channels with severe ISI and spectral zeros [2, Sec. 9.4]. A common approach for reduced-complexity MAP symbol detection is a simplification of the trellis search within the BCJR algorithm [4]. Either a reduced search on the full-complexity trellis can be performed (e.g., the M-BCJR algorithm [5]), or the number of trellis states can be reduced (e.g., the RS-BCJR algorithm [6]). However, these algorithms reduce the complexity of the BCJR algorithm only by a scalar factor and the performance-complexity tradeoff is only satisfactory for a particular subset of ISI channels [4]. An alternative approach to reduce the detection complexity is channel shortening [7]. Filtering the channel output with a channel shortening filter and then applying the BCJR algorithm on the shortened channel model enables a significantly reduced detection complexity but potentially comes with a performance-complexity tradeoff.

The advent of suboptimal iterative decoding in the context of turbo codes and LDPC codes has led to a rediscovery of message passing on graphical models. The powerful *factor graph* framework [8] provides a universal modeling tool for algorithms with controllable complexity. Based on the Ungerboeck observation model [9], Colavolpe *et al.* employed factor graphs and the SPA to derive a symbol detector with substantially reduced complexity [10]. In particular, the complexity is linear both in the block length and the channel memory. The proposed algorithm is however suboptimal, since its underlying factor graph contains cycles.

Recently, model-based deep learning has shown great potential to empower various suboptimal communication algorithms [11] and overcome their limitations. In [12], the factor

Manuscript received 31 March 2022; revised 8 August 2022; accepted 18 September 2022. Date of publication 26 September 2022; date of current version 18 November 2022. This work has received funding in part from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 101001899) and in part from the German Federal Ministry of Education and Research (BMBF) within the project Open6GHub (grant agreement 16KISK010). An earlier version of this paper was presented at the IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), Oulu, Finland, July 2022 [DOI: 10.1109/SPAWC51304.2022.9834025]. The associate editor coordinating the review of this article and approving it for publication was H.-C. Wu. *(Corresponding author: Luca Schmid.)*

The authors are with the Communications Engineering Laboratory (CEL), Karlsruhe Institute of Technology (KIT), 76187 Karlsruhe, Germany (e-mail: luca.schmid@kit.edu; schmalen@kit.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCOMM.2022.3209884.

Digital Object Identifier 10.1109/TCOMM.2022.3209884

nodes of a cycle-free factor graph are replaced by deep neural networks (DNNs) that are utilized to learn the local mappings of the factor nodes, thereby robustifying the algorithm towards model uncertainties. However, the algorithm still suffers from a complexity which is comparable to the complexity of the BCJR algorithm. Therefore, we focus on model-based deep learning approaches which are based on the Ungerboeck observation model, in the following. To mitigate the performance loss for cyclic factor graphs, a graph neural network (GNN), which is structurally identical to the original graph but has fully parametrized message updates, is proposed in [13]. The GNN runs conjointly to the original algorithm and corrects the SPA messages after each iteration. The authors in [14] compensate the performance degradation due to cycles in the graph by concatenating a supplemental neural network (NN)-based factor node (FN) to the factor graph. This additional FN is connected to all variable nodes (VNs) and is optimized in an end-to-end manner. However, the underlying NN structure is specifically tailored to binary transmission which substantially limits its scope of application. Instead of replacing different components of the factor graph by DNNs, the SPA is unfolded to a DNN and the resulting graph is equipped with tunable weights in [15]. This approach is known as neural belief propagation (NBP).

This paper aims at closing the gap between optimum and low-complexity symbol detection. We consider NBP on the Ungerboeck-based factor graph and further generalize the graph by introducing additional multiplicative weights within the FNs. Optimizing all weights in an end-to-end manner leads to considerable performance gains. Moreover, we leverage the high sensibility of the SPA to a variation of the underlying graph by applying an optimizable linear filter to the channel output, which allows us to modify the observation model and thereby the factor graph itself. By a combination of multiple factor graph instances in parallel, as well a dynamic variation of the graphs over the message passing iterations, we exploit this graph diversity in order to significantly improve the overall detection performance, and close the gap to optimum symbol detection for a variety of channels.

The remainder of this paper is organized as follows. In Section II, we briefly review the concept of factor graphs and the SPA [8]. Section III formulates the fundamental problem of symbol detection on channels with linear ISI. Using the factor graph framework, we present a suboptimal, but low-complexity, symbol detection algorithm. This provides the basis for Section IV, in which we propose and discuss various generalizations and enhancements to the algorithm. Section V examines the behavior of the proposed algorithms for different linear ISI channels and quantifies its performance compared to existing symbol detectors. Some concluding remarks are given in Section VI.

A. Notation

Throughout the paper, we use bold letters for non-scalar quantities. Upper case letters denote matrices $\boldsymbol{X}$ and $X_{m,n}$ represents the entry at row m and column n. Lower case letters are used for column vectors $\boldsymbol{x}$. The ith element of $\boldsymbol{x}$ is written as x_i and the stacking of a vector from multiple scalars is denoted by $[x_i]_{i=1}^{n} = \boldsymbol{x}$. $\|\cdot\|$ denotes the Euclidean norm and $(\cdot)^{\mathrm{H}}$ is the conjugate transpose (Hermitian) operator. The computation of the term $\ln\left(\mathrm{e}^{\delta_1} + \ldots + \mathrm{e}^{\delta_n}\right)$ can be carried out using the Jacobian logarithm [16] and is denoted by $\max_i{}^{\star}\, \delta_i$. The probability measure of a random variable (RV) x evaluated at x is denoted by $P_{\mathsf{x}}(\mathsf{x} = x)$. If it is clear from the context, we may use the shorter notation $P(\mathsf{x} = x)$ for the sake of simplicity. The probability density function (PDF) of a continuous RV y is denoted by $p_{\mathsf{y}}(y)$ or $p(y)$. The probability mass function (PMF) of a discrete RV x is $P_{\mathsf{x}}(x)$ or $P(x)$. The Gaussian distribution, characterized by its mean μ and variance σ^2, is written as $\mathcal{N}(\mu, \sigma^2)$. The expected value of an RV x is denoted by $\mathbb{E}_{\mathsf{x}}\{\mathsf{x}\}$ and the mutual information between the RVs x and y is $I(\mathsf{x}; \mathsf{y})$. We use calligraphic letters to denote a set $\mathcal{X}$ of cardinality $|\mathcal{X}|$.

II. Factor Graphs and Marginalization

The framework of factor graphs and the SPA is a flexible tool for algorithmic modeling of efficient inference algorithms. By representing the factorization of a composite global function of many variables in a graphical way, the computation of various marginalizations of this function can be efficiently implemented by a message passing algorithm. Since factor graphs are the foundation of our work, we review the basic concepts in this section.

Let $f(\mathcal{X})$ be a so-called "global" function which depends on a set of variables $\mathcal{X} = \{x_0, x_1, \ldots, x_n\}$ with $x_i \in \mathcal{M}$ for $i = 0, \ldots, n$. The marginalization of $f(\mathcal{X})$ towards a single marginal variable x_i typically requires $|\mathcal{M}|^n$ additions which can quickly become computationally infeasable for large n. The complexity can be significantly reduced by means of the distributive law if the global function is factorizable [8]. Let us assume that $f(\mathcal{X})$ can be factorized as

$$f(\mathcal{X}) = \prod_{j=1}^{J} f_j(\mathcal{X}_j), \quad \mathcal{X}_j \subset \mathcal{X}, \tag{1}$$

where each factor $f_j(\mathcal{X}_j)$ only depends on a subset of the variables $\mathcal{X}_j$. A *factor graph* represents the factorization of a multivariate function in a graphical way [17]. The following rules define the bijective relationship between the generic factorization in (1) and its corresponding factor graph:

- Every factor $f_j(\mathcal{X}_j)$ is represented by a unique vertex, the so-called FN f_j.
- Every variable $x \in \mathcal{X}$ is represented by a unique vertex, the so-called VN x.
- An FN f_j is connected to a VN x if and only if the corresponding factor $f_j(\mathcal{X}_j)$ is a function of x, i.e., if $x \in \mathcal{X}_j =: \mathcal{N}(f_j)$.

The notation $\mathcal{N}(f_j)$ denotes the *neighborhood* of the FN and is introduced to emphasize the fact that all variables on which a factor depends are represented by adjacent VNs in the factor graph. Equivalently, $\mathcal{N}(x) := \{f_j,\, j = 1, \ldots, J : x \in \mathcal{N}(f_j)\}$ denotes the neighborhood of the VN x. It is worth mentioning that the factor graph representation of a factorization is unique with respect to the structure of the resulting graph. However, there can be various factorizations of the same global function, leading to disparate factor graph representations [8].

The SPA is a message passing algorithm which computes the marginalization $f(x_i)$ of the global function $f(\mathcal{X})$ towards each variable $x_i \in \mathcal{X}$, respectively. It implicitly leverages the distributive law on the factorization of $f(\mathcal{X})$. Messages are propagated between the nodes of the factor graph along its edges and represent interim results of the marginalization. Let $\mu_{f_j \to x}(x)$ denote a message sent from FN f_j along an outgoing edge to VN x. Consequently, $\mu_{x \to f_j}(x)$ denotes a message on the same edge, but sent in the opposite direction. The message passing algorithm is based on one central message update rule for the VNs and FNs, respectively. In the logarithmic domain,[1] the message updates are

$$\mu_{x \to f_j}(x) = \sum_{f' \in \mathcal{N}(x) \setminus \{f_j\}} \mu_{f' \to x}(x) \tag{2}$$

$$\mu_{f_j \to x}(x) = \max_{\sim\{x\}}{}^{\star} \left(\ln(f_j(\mathcal{X}_j)) + \sum_{x' \in \mathcal{N}(f_j) \setminus \{x\}} \mu_{x' \to f_j}(x') \right). \tag{3}$$

They define the computation of an outgoing message, given the incoming messages on all *other* incident edges of a node [8], called *extrinsic messages*. The local marginalization in the FN to VN update, i.e., the Jacobian algorithm over all extrinsic variables, is thereby denoted by the *summary* operator $\max^{\star}{}_{\sim\{x\}}$.

Based on the SPA message update rule, we can compute marginals by propagating messages through the respective factor graph. If the graph is tree-structured, messages travel forward and backward through the entire graph, starting at the leaf nodes. Based on the computed messages, the exact marginals

$$f(x_i) = \exp\left(\sum_{f' \in \mathcal{N}(x_i)} \mu_{f' \to x_i}(x_i) \right)$$

can be obtained. Since the message updates are local [8] and because the SPA makes no reference to the topology of the graph [18], the SPA may also be applied to factor graphs with cycles, yielding an iterative algorithm. The messages are initialized with an unbiased state in iteration $n = 0$ and are iteratively updated by following a certain schedule until convergence or a stopping criterion is reached. In the case of cyclic factor graphs, the superscript $^{(n)}$, with $n = 0, \ldots, N$, indicates the iteration in which the message $\mu_{a \to b}^{(n)}$ is computed. In general, convergence of the SPA on cyclic factor graphs is not guaranteed and the iterative algorithm only yields an approximation

$$\hat{f}(x_i) := \exp\left(\sum_{f' \in \mathcal{N}(x_i)} \mu_{f' \to x_i}^{(N)}(x_i) \right) \tag{4}$$

of the exact marginal $f(x_i)$ [8]. However, many successful applications, e.g., decoders of error-correcting codes, are based on message passing algorithms on cyclic graphs.

[1]When it comes to hardware implementation, it can be advantageous to carry out the SPA in the logarithmic domain due to less numerical instabilities and a reduced computational complexity.

III. Symbol Detection

We consider the transmission of an information sequence $\mathbf{c} = [\mathsf{c}_k]_{k=1}^{K} \in \mathcal{M}^K$ of a multilevel constellation $\mathcal{M} = \{\mathrm{m}_i \in \mathbb{C}, i = 1, \ldots, M\}$ over a complex baseband channel, impaired by linear interference and AWGN. The bit pattern of length $m := \log_2(M)$ which corresponds to a symbol c_k is denoted by $\mathbf{b}(\mathsf{c}_k) = [\mathsf{b}_i(\mathsf{c}_k)]_{i=1}^{m}$. The relationship between the independent and identically distributed (i.i.d.) information symbols c_k and the receive symbols y_k can be expressed by an equivalent discrete-time channel model [19]:

$$\mathsf{y}_k = \sum_{\ell=0}^{L} h_\ell \mathsf{c}_{k-\ell} + \mathsf{w}_k. \quad k = 1, \ldots, K+L. \tag{5}$$

For a channel with memory L, $\boldsymbol{h} \in \mathbb{C}^{L+1}$ is the finite channel impulse response and $\mathbf{w} \sim \mathcal{CN}(0, \sigma^2 \boldsymbol{I})$ denotes white circular Gaussian noise. The channel is assumed to be static, which leads to the channel impulse response $\boldsymbol{h}$ being constant over time. The symbols c_k for $k < 1$ and $k > K$ are information symbols from the same constellation $\mathcal{M}$. We assume that these boundary symbols are fully known at the receiver, as they are either pilot symbols or information symbols from an adjacent transmission block which was already detected and decoded. An equivalent transmit sequence is given by $\check{\mathbf{c}} := [\mathsf{c}_k]_{k=1-L}^{K+L} \in \mathcal{M}^{(K+2L)}$. Since the interference is linear, (5) can be described in matrix vector notation:

$$\mathbf{y} = \boldsymbol{H} \check{\mathbf{c}} + \mathbf{w}.$$

The matrix $\boldsymbol{H} \in \mathbb{C}^{(K+L) \times (K+2L)}$ is a band-structured Toeplitz matrix which represents the convolution of the transmit sequence $\check{\mathbf{c}}$ with the channel impulse response $\boldsymbol{h}$.

We study the problem of symbol detection, i.e., the estimation of the information symbols c_k, $k = 1, \ldots, K$ from a sequence $\mathbf{y}$, observed at the receiver. In the context of Bayesian inference, we are interested in the a posteriori probability (APP) distribution

$$P(\boldsymbol{c}|\boldsymbol{y}) \propto p(\boldsymbol{y}|\boldsymbol{c}) P(\boldsymbol{c}),$$

where proportionality $\propto$ denotes two terms only differing in a factor independent of $\mathbf{c}$. The APP can be factored into the likelihood

$$p(\boldsymbol{y}|\boldsymbol{c}) = \frac{1}{(\pi\sigma^2)^K} \exp\left(-\frac{\|\boldsymbol{y} - \boldsymbol{H}\boldsymbol{c}\|^2}{\sigma^2} \right) \tag{6}$$

and the a priori probability $P(\boldsymbol{c})$, using Bayes' theorem [20, Chap. 2]. The symbol-wise APPs are obtained by computing the marginals

$$P(\mathsf{c}_k = c|\boldsymbol{y}) = \sum_{\substack{\boldsymbol{c} \in \mathcal{M}^K \\ c_k = c}} P(\mathbf{c} = \boldsymbol{c}|\boldsymbol{y}), \quad k = 1, \ldots, K, \tag{7}$$

on which symbol detection can be based. In case of hard decision, the symbol-wise MAP detection

$$\hat{\mathsf{c}}_k = \operatorname*{argmax}_{c \in \mathcal{M}} P(\mathsf{c}_k = c|\boldsymbol{y}), \quad k = 1, \ldots, K$$

yields the minimum probability of error for each symbol decision, respectively [2, Sec. 9.3].

A. Factor Graph Modeling

The computation of the symbol-wise APPs $P(\mathtt{c}_k = c|\boldsymbol{y})$ in (7) requires K marginalizations which we can efficiently compute by employing the factor graph framework. To model a factor graph, we need to find an appropriate factorization of the APP distribution $P(\boldsymbol{c}|\boldsymbol{y})$. The likelihood in (6) can be expressed as [10]

$$p(\boldsymbol{y}|\boldsymbol{c}) \propto \exp\left(\frac{2\mathrm{Re}\{\boldsymbol{c}^{\mathrm{H}}\boldsymbol{H}^{\mathrm{H}}\boldsymbol{y}\} - \boldsymbol{c}^{\mathrm{H}}\boldsymbol{H}^{\mathrm{H}}\boldsymbol{H}\boldsymbol{c}}{\sigma^2}\right).$$

We substitute

$$\boldsymbol{G} := \boldsymbol{H}^{\mathrm{H}}\boldsymbol{H}, \qquad \boldsymbol{x} := \boldsymbol{H}^{\mathrm{H}}\boldsymbol{y} \tag{8}$$

and interpret $\boldsymbol{x}$ as an alternative observation at the receiver. This is commonly known as the *Ungerboeck observation model* [9]. By using

$$\boldsymbol{c}^{\mathrm{H}}\boldsymbol{x} = \sum_{k=1}^{K} x_k c_k^{\star}$$

$$\boldsymbol{c}^{\mathrm{H}}\boldsymbol{G}\boldsymbol{c} = \sum_{k=1}^{K} G_{k,k}|c_k|^2 - \sum_{k=1}^{K}\sum_{\substack{\ell=1\\ \ell\neq k}}^{K} \mathrm{Re}\{G_{k,\ell}c_\ell c_k^{\star}\},$$

the likelihood function can be factorized as

$$p(\boldsymbol{y}|\boldsymbol{c}) \propto \prod_{k=1}^{K}\left[F_k(c_k)\prod_{\substack{\ell=1\\ \ell\neq k}}^{K} J_{k,\ell}(c_k, c_\ell)\right]$$

with the factors

$$F_k(c_k) := \exp\left(\frac{1}{\sigma^2}\mathrm{Re}\{2\ x_k c_k^{\star} - G_{k,k}|c_k|^2\}\right) \tag{9}$$

$$J_{k,\ell}(c_k, c_\ell) \ = \ \exp\left(-\frac{1}{\sigma^2}\mathrm{Re}\{G_{k,\ell}c_\ell c_k^{\star}\}\right). \tag{10}$$

The factors $J_{k,\ell}(c_k, c_\ell)$ and $J_{\ell,k}(c_\ell, c_k)$ depend on the same variables and can thus be condensed to one factor

$$I_{k,\ell}(c_k, c_\ell) := \ J_{k,\ell}(c_k, c_\ell)J_{\ell,k}(c_\ell, c_k), \quad k > \ell \tag{11}$$

$$= \ J_{k,\ell}^2(c_k, c_\ell), \tag{12}$$

where (12) exploits the Hermitian symmetry of $\boldsymbol{G}$. The factor $I_{k,\ell}(c_k, c_\ell)$ is symmetric with respect to k and ℓ, i.e., $I_{k,\ell}(c_k, c_\ell) = I_{\ell,k}(c_\ell, c_k)$. The a priori distribution

$$P(\boldsymbol{c}) = \prod_{k=1}^{K} P(c_k)$$

can be factorized due to the statistical independence of the information symbols. In summary, the APP can be expressed in the factorization

$$P(\boldsymbol{c}|\boldsymbol{y}) \propto \prod_{k=1}^{K} P(c_k) \prod_{k=1}^{K}\left[F_k(c_k)\prod_{\ell<k} I_{k,\ell}(c_k, c_\ell)\right], \tag{13}$$

which is represented by a factor graph in Fig. 1. Applying the SPA on this factor graph to develop a symbol detection algorithm was first proposed by Colavolpe *et al.* in [10]. We will refer to this algorithm as UFG (Ungerboeck-based factor graph symbol detector) in the following. We initialize all messages with $\mu_0(c_k) := -\ln(M)$ and perform N SPA iterations on the graph. We apply a flooding schedule, i.e., one iteration comprises the simultaneous update of all messages from VNs to FNs in a first step, followed by the update of messages propagating in the opposite direction in a second step. The soft output $\hat{P}(c_k|\boldsymbol{y})$ is finally obtained by applying (4) to all VNs.

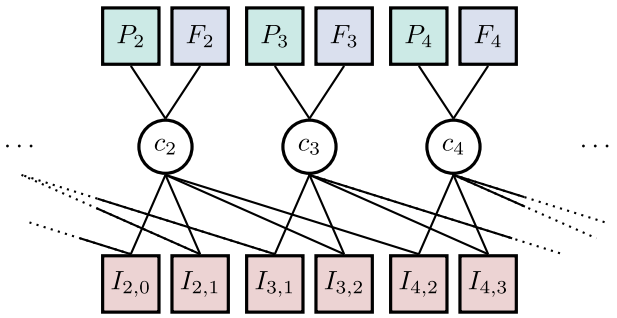

Fig. 1. Factor graph representation of (13) for $L = 2$.

The complexity of factor graph-based algorithms can be estimated by considering the number of FNs and their node degree, since the FN update rule (3) is computationally more demanding than the operation (2) at the VNs [21]. The UFG symbol detector is based on a factor graph with maximum FN degree of 2 of the $I_{k,\ell}$ nodes. Therefore, the algorithm has a computational complexity which only grows linearly with the channel memory L. This makes the UFG algorithm an attractive low-complexity alternative to the well established BCJR algorithm which has an exponentially growing complexity with L.

Although the factorization (13) is exact, the UFG algorithm only yields an approximation for the symbol-wise APPs due to cycles within the underlying factor graph. It is thus a suboptimal algorithm. By agglomerating variable nodes, the cycles within the factor graph can be eliminated. This leads to a forward-backward algorithm described in [22], yielding the exact marginalization $P(c_k|\boldsymbol{y})$, on which optimum MAP detection can be carried out. However, the appealing complexity properties of the cyclic factor graph vanish if clustering is applied: the SPA algorithm on the clustered factor graph has a complexity similar to the BCJR algorithm [22], which grows exponentially in both channel memory L and number of bits per symbol m.

IV. Neural Enhancement of Factor Graph-Based Symbol Detection

Driven by the appealing complexity properties of the UFG algorithm, we urge to compensate for its suboptimality by neurally enhancing both the factor graph and the SPA. In particular, we consider NBP and an optimization of the FNs in Sec. IV-A. We further propose a dynamic factor graph transition in Sec. IV-B, specifically tailored to this particular problem. Based thereupon, we present a novel symbol detection algorithm, which is formally defined in Sec. IV-C. Section IV-D details the parameter optimization and introduces the bitwise mutual information (BMI) as an objective function for optimization and evaluation.

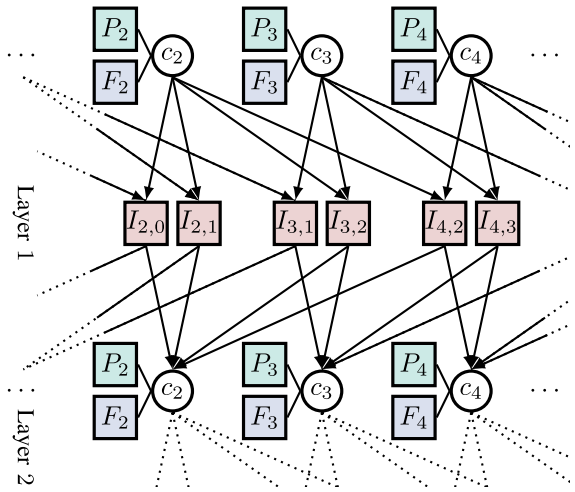

Fig. 2. Unrolled SPA on the factor graph of Fig. 1: each layer corresponds to one iteration of the SPA.

A. Neural Belief Propagation and FN Enhancement

Applying the SPA to cyclic factor graphs yields an iterative algorithm. By the use of deep unfolding, first introduced in [23], an iterative algorithm can be converted into a DNN. If a flooding schedule is applied to the factor graph in Fig. 1, a single iteration of the SPA consists of propagating messages from VNs to FNs and back. Unfolding the N iterations of the SPA on the factor graph is thus natural since each iteration is already (factor) graph-based. The resulting unrolled network comprises N layers and is shown in Fig. 2. Messages are propagated through the DNN in a feed-forward fashion. For the sake of simplicity, we introduce a shorter notation for the messages in the DNN:

$$\mu_{k,j}^{(n)}(c_k) := \mu_{c_k \to I_{k,k+j}}^{(n)}(c_k)$$
$$\nu_{k,j}^{(n)}(c_k) := \mu_{I_{k,k+j} \to c_k}^{(n)}(c_k).$$

Due to the inherent band structure of $\boldsymbol{G}$, the network is not fully connected and the index j is limited to $j \in \mathcal{J} := \{-L, \dots, -1, 1, \dots, L\}$. VNs and FNs accept incoming messages from the previous layer and apply the SPA message update rule. The resulting outgoing messages are forwarded downstream to the next layer. As a consequence, each transmitted message in every iteration has its individually assigned edge. By accordingly weighting each message/edge, we attempt to mitigate the effects of short cycles and improve the detection performance compared to the UFG algorithm. Optimizing the weights towards a loss function by the use of established deep learning techniques is known as NBP, first introduced in the context of belief propagation (BP) on Tanner graphs for decoding of linear block codes [15].

We limit the set of weights to the edges between VNs c_k and FNs $I_{k,\ell}$. The FNs P_k and F_k have degree 1. Incident edges are thus not included in the cycles of the factor graph and are consequently not weighted. In consistency with the message notation, the message $\mu_{k,j}^{(n)}(c_k)$ is multiplied by the weight $w_{\mathrm{v},k,j}^{(n)}$. Vice versa, messages from FNs to VNs $\nu_{k,j}^{(n)}(c_k)$ are weighted by $w_{\mathrm{f},k,j}^{(n)}$. We further generalize the FNs of the UFG algorithm by the application of multiplicative weights $\kappa_{i,k}^{(n)}$ and $\lambda_{k,\ell}^{(n)}$ within the FN computation, in order to increase the parameter optimization space further. The generalized factors, given in the logarithmic domain, are

$$\tilde{F}_k^{(n)}(c_k) := \frac{\kappa_{1,k}^{(n)}}{\sigma^2} \mathrm{Re}\Big\{ \kappa_{2,k}^{(n)} 2\, x_k c_k^* - \kappa_{3,k}^{(n)} G_{k,k} |c_k|^2 \Big\}$$
$$\tilde{I}_{k,\ell}^{(n)}(c_k, c_\ell) := \lambda_{k,\ell}^{(n)} \Big(\tilde{J}_{k,\ell}(c_k, c_\ell) + \tilde{J}_{\ell,k}(c_\ell, c_k) \Big)$$
$$\tilde{J}_{k,\ell}(c_k, c_\ell) := \ln(J_{k,\ell}(c_k, c_\ell)) = -\frac{1}{\sigma^2} \mathrm{Re}\{ G_{k,\ell} c_\ell c_k^\star \}.$$

One central motivation to introduce the (trainable) weights $\lambda_{k,\ell}^{(n)}$ and $\kappa_{1,k}^{(n)}$ is to artificially attenuate the $1/\sigma^2$ term inside the FNs. By adopting a value of $1/\sigma^2$ in the factor graph smaller than the actual one, the overconfidence of the SPA messages can be reduced, by describing the channel as if it added more noise than it actually does [10]. In summary, the set of parameters for the generalized algorithm is

$$\mathcal{P}_{\text{NBP}} := \Big\{ w_{\mathrm{v},k,j}^{(n)}, w_{\mathrm{f},k,j}^{(n)}, \kappa_{i,k}^{(n)}, \lambda_{k,\ell}^{(n)},\ n = 1, \dots, N,\ j \in \mathcal{J},\ i = 1, 2, 3,\ k = 1, \dots, K \Big\}$$

and contains $|\mathcal{P}_{\text{NBP}}| = NK(5L + 3)$ real-valued elements. Note that we define all parameters of $\mathcal{P}_{\text{NBP}}$ to be independent of c_k. For instance, we constrain the weights $w_{\mathrm{v},k,j}^{(n)}$ and $w_{\mathrm{f},k,j}^{(n)}$ to be scalars although the SPA messages $\mu_{k,j}^{(n)}(c_k)$ and $\nu_{k,j}^{(n)}(c_k)$ are M-dimensional vectors. This limitation significantly reduces the total number of parameters which need to be optimized. The UFG algorithm is a special instance of the proposed generalization and is obtained by the parametrization $\mathcal{P}_{\text{NBP}} = \mathcal{P}_1 := \{1, \dots, 1\}$. By optimizing $\mathcal{P}_{\text{NBP}}$, the performance of the resulting algorithm can thus not be inferior to the UFG algorithm, but might yield a performance gain [15].

Note that our approach of directly enhancing the UFG algorithm by generalizing its underlying graph and weighting the SPA messages is conceptually different from neural augmentation techniques such as GNNs [13]. Neural augmentation does not modify the model-based algorithm directly but instead utilizes an external DNN to correct the SPA messages of the original algorithm in each iteration [11].

B. Dynamic Factor Graph Transition

One key principle of the SPA is the extrinsic information rule. By computing an outgoing message only based on incoming messages of extrinsic edges, the SPA ensures that only "new information" is propagated through the graph. The extrinsic concept is violated in a cyclic factor graph, where messages propagate within a loop repetitively and intrinsic information is mistaken for extrinsic information by the nodes involved in the cycle. This violation is inevitable if a sufficient number of iterations is performed on a cyclic factor graph.

In Sec. IV-A, we have introduced a generalization of the FNs as well as NBP in order to mitigate the performance degradation due to the cycles. However, inherent cycles still exist in the unfolded architecture of the NBP and the proposed methods might not be able to fully compensate for their existence. Consequently, we propose an additional strategy to reduce the effect of cycles in a more intrinsic way.

By dynamically modifying the underlying factor graph on which the message passing is iteratively performed, messages do not repeatedly arrive at one and the same factor node because either the graph structure and/or the FNs therein have changed. In specific, we propose to periodically change the factor graph's underlying observation model by the application of a linear filter. The channel output $\mathbf{y}$ is therefore preprocessed by a finite impulse response (FIR) filter and the result $\mathbf{x} = \boldsymbol{P}\mathbf{y}$ is then used as a new observation for the inference task. $\boldsymbol{P}$ is a band-structured convolutional Toeplitz matrix based on the generic impulse response $\boldsymbol{p} \in \mathbb{C}^{L_\mathrm{p}+1}$ of the FIR preprocessing filter.

The factorization in (13) represents the APP and is thus optimal in the context of Bayesian inference. In order to maintain this optimality, the output of the preprocessor $\mathbf{x}$ must be a *sufficient statistic* for the estimation of $\mathbf{c}$ [24]. Only in this case, the data processing inequality holds with equality [25, Sec. 2.8] and the original observation $\mathbf{y}$ is irrelevant for the MAP detection, if $\mathbf{x}$ is available [10]. In fact, the original UFG algorithm implicitly uses a preprocessor in (8) by applying a matched filter $\boldsymbol{P} = \boldsymbol{H}^\mathrm{H}$ to the channel output $\boldsymbol{y}$. Based on this Ungerboeck observation model, inference is carried out using the new observation $\boldsymbol{x} = \boldsymbol{H}^\mathrm{H}\boldsymbol{y}$. In order to generate a multitude of different factor graphs, we generalize the observation model

$$\tilde{\boldsymbol{x}} := \boldsymbol{P}\boldsymbol{y}, \qquad \tilde{\boldsymbol{G}} := \boldsymbol{P}\boldsymbol{H}.$$

Note that varying $\tilde{\boldsymbol{G}}$ directly affects the underlying factor graph in (9) and (10). The matrix $\tilde{\boldsymbol{G}}$ is in general not Hermitian symmetric. Consequently, the simplification in (12) is not valid and the factor $I_{k,\ell}$ is defined by (11). This, however, does not change the structure of the underlying factor graph but only of the FNs therein. By using different preprocessors, we gain distinct factor graph instances. We can leverage multiple factor graph instances in two dimensions:

- **Dynamic factor graph transition**: Instantiate S different factor graphs, so-called *stages*. Perform N' SPA iterations on one stage s, before proceeding to the next stage $s+1$.
- **Parallelism**: In each stage, apply B distinct factor graph instances in parallel. Combine the B results after each stage s to improve the quality of the APP estimation.

Based on this idea, we propose the novel symbol detection algorithm GAP (graph alteration by preprocessing). Figure 3 summarizes the information flow of the GAP algorithm. Each of the $S \cdot B$ factor graph instances uses its individual preprocessor $\boldsymbol{P}^{(s,b)}$, $s = 1, \ldots, S$, $b = 1, \ldots, B$ and thus its individual observation model $\boldsymbol{x}^{(s,b)} = \boldsymbol{P}^{(s,b)}\boldsymbol{y}$. The B factor graphs of a stage s work independently and in parallel. Each factor graph performs N' SPA iterations, based on $\boldsymbol{x}^{(s,b)}$. To merge the B individual results $\hat{P}_{s,b}(c_k|\boldsymbol{y})$, we combine the logarithmic APP distributions of the symbol detectors by addition, followed by a normalization:

$$\ln\Big(\hat{P}_s(c_k|\boldsymbol{y})\Big) = \sum_{b=1}^{B} \ln\Big(\hat{P}_{s,b}(c_k|\boldsymbol{y})\Big) - \max_{\mathrm{m}\in\mathcal{M}}{}^{\star}\left(\sum_{b=1}^{B} \ln\Big(\hat{P}_{s,b}(c_k = \mathrm{m}|\boldsymbol{y})\Big)\right). \quad (14)$$

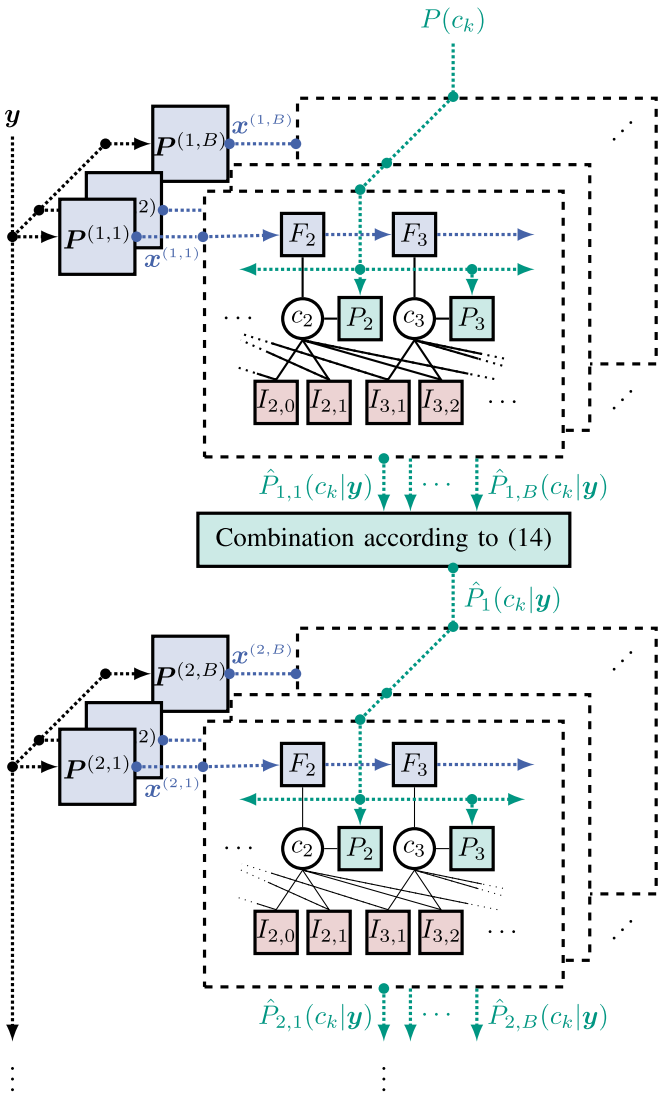

Fig. 3. Hierarchical structure of the GAP algorithm with B branches and S stages for a channel with memory $L = 2$. The GAP algorithm accepts statistical a priori information $P(c_k)$ about the information symbols c_k as well as the channel observation $\boldsymbol{y}$. It returns an estimation of the symbol-wise APP distributions $\hat{P}_S(c_k|\boldsymbol{y})$.

The idea of using multiple parallel processors of varying behavior in order to increase overall performance was already proposed in the context of channel decoding on Tanner graphs, e.g., in [26]. Our method, however, differs in the way of how to combine the information of the parallel branches. Instead of selecting the most reliable output of all branches or uniformly averaging over the probabilities, both proposed in [26], our approach in (14) inherently weights the prioritization of the individual results based on the parametrization $\mathcal{P}_\mathrm{NBP}$.

The combined result $\hat{P}_s(c_k|\boldsymbol{y})$ is passed to the subsequent stage, by setting the FNs $P_k^{(n)}$ of the factor graphs in stage $s+1$ and branch b to

$$P_k^{(n)} = \exp\left(w_{\mathrm{p},s,b}^{(n)} \cdot \ln\left(\hat{P}_s(c_k|\boldsymbol{y})\right)\right), \quad n = 1, \ldots, N'.$$

The parameters $w_{\mathrm{p},s,b}^{(n)}$ are introduced to control the dependency between adjacent stages. Note that $\hat{P}_s(c_k|\boldsymbol{y})$ is only an (imprecise) approximation of the symbol-wise

Algorithm 1 GFG

Data: $\boldsymbol{y}$, $P_{\mathrm{e}}(c_k)$, $\mathcal{P}_{\mathrm{p}}$, $\mathcal{P}_{\mathrm{NBP}}$, $\boldsymbol{P}$, N

1 Preprocessing $\tilde{\boldsymbol{x}} = \boldsymbol{P}\boldsymbol{y}$

2 Initialize messages $\nu_{k,j}^{(0)}(c_k) = -\ln(M)$

3 **for** $n = 1, \ldots, N$ **do**

4 $\quad \mu_{k,j}^{(n)}(c_k) = w_{\mathrm{v},k,j}^{(n)} \left(\xi_k^{(n)}(c_k) + \sum_{j' \in \mathcal{J} \setminus j} \nu_{k,j'}^{(n-1)}(c_k) \right)$

5 $\quad \nu_{k,j}^{(n)}(c_k) = w_{\mathrm{f},k,j}^{(n)} \max_{c_{k+j}}{}^{\star} \left(\tilde{I}_{k,k+j}^{(n)}(c_k) + \mu_{k+j,-j}^{(n)}(c_k) \right)$

6 Compute symbol-wise APP estimates

$$\hat{P}(c_k|\boldsymbol{y}) = \exp\left(\xi_k^{(N+1)}(c_k) + \sum_{j' \in \mathcal{J}} \nu_{k,j'}^{(N)}(c_k) \right)$$

Result: $\hat{P}(c_k|\boldsymbol{y}), \quad k = 1, \ldots, K$

Algorithm 2 GAP (S, B, N')

Data: $\boldsymbol{y}$, $P(c_k)$, $\mathcal{P}_{\mathrm{GAP}}$

1 Initialize extrinsic information $\hat{P}_0 = P(c_k)$

2 **for** $s = 1, \ldots, S$ **do**

3 $\quad$ **for** $b = 1, \ldots, B$ **do**

4 $\quad\quad \hat{P}_{b,s}(c_k) =$ $\mathrm{GFG}_{b,s}\left(\boldsymbol{y}, \hat{P}_{s-1}(c_k), \mathcal{P}_{\mathrm{p}}^{(s,b)}, \mathcal{P}_{\mathrm{NBP}}^{(s,b)}, \boldsymbol{P}^{(s,b)}, N' \right)$

5 $\quad$ Merge APP estimates $\hat{P}_{b,s}(c_k)$ of all branches $b = 1, \ldots B$ into $\hat{P}_s(c_k)$ according to (14).

Result: Symbol-wise APP estimates

$$\hat{P}(c_k|\boldsymbol{y}) = \hat{P}_S(c_k), \quad k = 1, \ldots, K$$

APP distributions $P(c_k|\boldsymbol{y})$. Finding a more precise approximation is the objective of the subsequent stages, respectively. Therefore, the weights $w_{\mathrm{p},s,b}^{(n)}$ can dampen the influence of $\hat{P}_s(c_k|\boldsymbol{y})$, e.g., in the final iterations $n \in N'$. We summarize the introduced parameters for each stage s and branch b in

$$\mathcal{P}_{\mathrm{p}}^{(s,b)} := \left\{ w_{\mathrm{p},s,b}^{(n)},\, n = 1, \ldots, N' \right\}.$$

Within a specific factor graph instance with fixed s and b, we use the shorter notation $w_{\mathrm{p}}^{(n)}$.

C. Algorithm and Complexity

We formalize the discussed methods for the neural enhancement of factor graph-based symbol detection. Algorithm 1 defines the GFG (generalized factor graph-based detection) algorithm which generalizes the UFG algorithm by NBP, neurally enhanced FNs as well as a generalized observation model. The GFG algorithm is parametrized by $\mathcal{P}_{\mathrm{NBP}}$, $\mathcal{P}_{\mathrm{p}}$ and the preprocessor $\boldsymbol{P} \in \mathbb{C}^{(K+2L,K+L)}$, which is assumed to be a band-structured Toeplitz matrix and describes the convolution with an FIR filter $\boldsymbol{p} \in \mathbb{C}^{L_{\mathrm{p}}+1}$. The algorithm accepts the channel output $\boldsymbol{y} \in \mathbb{C}^{K+L}$ as well as extrinsic information $P_{\mathrm{e}}(c_k)$ of the information symbols c_k, e.g., statistical a priori knowledge $P_{\mathrm{e}}(c_k) = P(c_k)$. The preprocessor is applied to the channel observation and all messages from FNs $I_{k,\ell}(c_k, c_\ell)$ to VNs c_k are initialized to the same value. According to the SPA, messages of degree 1 FNs do not receive extrinsic information and are consequently not updated. The messages from the FNs $\tilde{F}_k(c_k)$ and $P_k(c_k)$ to the VNs c_k can thus be computed in advance and are summarized in

$$\xi_k^{(n)}(c_k) := w_{\mathrm{p}}^{(n)} \cdot \ln(P_{\mathrm{e}}(c_k)) + \tilde{F}_k^{(n)}(c_k).$$

Subsequently, messages are passed iteratively between FNs and VNs based on the SPA. The message update in line 5 is simplified due to the degree 2 nature of the FNs $\tilde{I}_{k,\ell}$. The symbol-wise APP estimates $\hat{P}(c_k|\boldsymbol{y})$ are computed by a final marginalization for $k = 1, \ldots, K$ and are the result of the GFG algorithm.

Based on the dynamic factor graph transition discussed in Sec. IV-B, we formally define the novel symbol detection algorithm GAP. The GAP algorithm is a hierarchical detection algorithm, structured in S stages and B branches as illustrated in Fig. 3. Each of the $B \cdot S$ units can be seen as an individual GFG symbol detector which is characterized by a unique parametrization $\mathcal{P}_{\mathrm{NBP}}^{(s,b)}$, $\mathcal{P}_{\mathrm{p}}^{(s,b)}$ and an individual preprocessor $\boldsymbol{P}^{(s,b)} \in \mathbb{C}^{(K+2L,K+L)}$. The GAP algorithm is defined in Algorithm 2. It accepts the channel output $\boldsymbol{y}$, statistical a priori knowledge $P(c_k)$ about the information symbols as well as the parametrization

$$\mathcal{P}_{\mathrm{GAP}} := \bigcup_{s=1}^{S} \bigcup_{b=1}^{B} \mathcal{P}_{\mathrm{NBP}}^{(s,b)} \cup \mathcal{P}_{\mathrm{p}}^{(s,b)} \cup \left\{ \boldsymbol{P}^{(s,b)} \right\},$$

where $\mathcal{P}_{\mathrm{NBP}}^{(s,b)}$ denotes the set $\mathcal{P}_{\mathrm{NBP}}$ for each individual GFG unit in stage s and branch b. The parameter set contains $|\mathcal{P}_{\mathrm{GAP}}| = SB(|\mathcal{P}_{\mathrm{NBP}}| + N + 2(L_{\mathrm{p}} + 1))$ real-valued elements. Note that all GFG units in one stage are fed by the same extrinsic information $P_{\mathrm{e}}(c_k)$ and only differ in their individual parametrization and preprocessor.

Concerning the computational complexity, the GAP algorithm shares the appealing properties of the UFG algorithm. Due to a constant node degree 2 of the FNs $\tilde{I}_{k,\ell}(c_k)$, the complexity grows linearly with the channel memory L, and quadratically with the size M of the constellation alphabet $\mathcal{M}$. Depending on the number of branches and stages, the GAP algorithm has an order of complexity $O(SBN'KLM^2)$. In transmission scenarios over channels with large memory L or high-order constellations, the GAP algorithm becomes a low-complexity alternative to the BCJR algorithm which has an asymptotic complexity of $O(KM^{L+1})$. For a detailed complexity comparison, Table I reports the number of real additions (ADD), real multiplications (MULT) and $\max^\star$ operations for the algorithms UFG, GAP as well as for the BCJR algorithm. One-time operations for initialization that are independent of the channel observation as well as boundary effects are neglected. Table II evaluates the complexity for a selection of specific transmission scenarios with a given channel memory L and constellation size M. The complexity parameter X is defined to be the total number of operations, comprising real additions, multiplications and $\max^\star$ operations that are required to estimate the APP of one symbol c_k. Note that the given parametrizations for (S, B, N') are of examplary nature and need to be specifically adapted to different channels in

TABLE I
NUMBER OF OPERATIONS PER INFORMATION SYMBOL FOR DIFFERENT SYMBOL DETECTORS OPERATING IN THE LOGARITHMIC DOMAIN

Algorithm	ADD	MULT	max*
BCJR	$8M^{L+1}$	$2M^{L+1}$	$3M^{L+1}$
UFG	$M(2L+4+N(6L+1))+2L$	$2(L+M+1)$	$2NLM^2$
GAP	$BS[2L_\mathrm{p}+M[N'(6L_\mathrm{p}+4)+(2L_\mathrm{p}+1)+M]+SM$	$BS(2L_\mathrm{p}+1)(2N'M+1)$	$MS[2BN'LM+1]$

TABLE II
COMPLEXITY PARAMETER X FOR $N=10$ AND DIFFERENT CONFIGURATIONS OF THE CONSTELLATION SIZE M AND THE CHANNEL MEMORY L

Algorithm		BPSK ($M=2$)		16-QAM ($M=16$)	
	(S,B,N')	$L=4$ $L_\mathrm{p}=9$	$L=10$ $L_\mathrm{p}=10$	$L=4$ $L_\mathrm{p}=9$	$L=10$ $L_\mathrm{p}=10$
BCJR		416	26624	$\approx 10^7$	$\approx 10^{14}$
UFG		866	2114	24722	61418
GAP	(1, 1, 10)	2723	3011	62069	68825
GAP	(5, 2, 4)	11460	12660	252610	279850

practice. See Sec. V for realistic parametrizations on some specific channels.

D. Parameter Optimization

The parameters in $\mathcal{P}_\mathrm{GAP}$ are jointly optimized towards an objective function in an end-to-end manner. Since the GAP algorithm embodies a NN, we rely on a rich pool of advanced optimization and training methods developed for feed-forward neural networks in the last years. For training, we use the Adam algorithm [27]; a stochastic gradient descent optimizer. The gradient can be computed using backpropagation [28], which is a standard method for NNs. All weights $\mathcal{P}_\mathrm{NBP}^{(s,b)}$ and $\mathcal{P}_\mathrm{p}^{(s,b)}$ are initialized with 1.0. The initial impulse responses $\boldsymbol{p}$ of the preprocessors $\boldsymbol{P}^{(s,b)}$ are independently sampled from a standard normal distribution.

We optimize the parametrization towards a maximum achievable rate between the channel input and the detector output. Many practical transmission systems use bit-interleaved coded modulation (BICM), which decouples the symbol detection from a binary soft-decision forward error correction (FEC) [29]. In BICM, the symbol detector soft output $\hat{P}(c_k|\boldsymbol{y})$ is converted by a bit-metric decoder (BMD) to binary soft information

$$\hat{P}(\mathsf{b}_i(\mathsf{c}_k)=b|\boldsymbol{y}) = \sum_{c\in\mathcal{M}_i^{(b)}} \hat{P}(\mathsf{c}_k=c|\boldsymbol{y}), \quad b\in\{0,1\}$$

with $\mathcal{M}_i^{(b)} := \{c\in\mathcal{M} : b_i(c)=b\}$. The resulting bit-wise APPs are typically expressed in log-likelihood ratios (LLRs)

$$L_{k,i}(\boldsymbol{y},\alpha) := \alpha \ln\left(\frac{\hat{P}(\mathsf{b}_i(\mathsf{c}_k)=0|\boldsymbol{y})}{\hat{P}(\mathsf{b}_i(\mathsf{c}_k)=1|\boldsymbol{y})}\right), \quad \alpha\in\mathbb{R}_{>0}.$$

If the LLR is based on suboptimal detection, i.e., if $\hat{P}(\mathsf{b}_i=b|\boldsymbol{y})$ is not the true APP, the scaling factor α corrects a potential LLR mismatch [30, Chap. 7]. After interleaving, the LLRs are fed to a bit-wise soft-decision FEC. By interpreting the BMD as a mismatched detector, the BMI is an achievable information rate for BICM [29]. The calculation of the BMI, detailed in [31], considers the BMD by assuming m parallel sub-channels transmitting on a binary basis instead of one symbol-based channel. Assuming i.i.d. transmit bits, the BMI[2] is defined as the sum of mutual informations $I(\mathsf{b}_i;\mathsf{y})$ of m unconditional bit-wise channel transmissions:

$$\begin{aligned}\mathrm{BMI} &:= \sum_{i=1}^{m} I(\mathsf{b}_i(\mathsf{c}_k);\mathsf{y}) \\ &= \sum_{i=1}^{m} \mathbb{E}_{\mathsf{b}_i,\mathsf{y}}\left\{\log_2\left(\frac{P_{\mathsf{b}_i(\mathsf{c}_k)|\mathsf{y}}(\mathsf{b}_i|\mathsf{y})}{P_{\mathsf{b}_i}(\mathsf{b}_i(\mathsf{c}_k))}\right)\right\}.\end{aligned}$$

By a sample mean estimation over D labeled data batches $\mathcal{D} := \{(\boldsymbol{c}^{(\mathrm{d})},\boldsymbol{y}^{(\mathrm{d})})_i : \boldsymbol{c}^{(\mathrm{d})}\in\mathcal{M}^K, \boldsymbol{y}^{(\mathrm{d})} = \boldsymbol{H}\boldsymbol{c}^{(\mathrm{d})}+\boldsymbol{w}_i, i=1,\dots,D\}$ and by assuming uniformly distributed information bits, a feasible approximation

$$\begin{aligned}\mathrm{BMI} \approx\ & \log_2(M) \\ & -\frac{1}{DK}\sum_{i=1}^{m}\sum_{k=0}^{K-1}\sum_{(\boldsymbol{c}^{(\mathrm{d})},\boldsymbol{y}^{(\mathrm{d})})\in\mathcal{D}} \\ & \times \log_2\Big(\exp\Big(-(-1)^{b_{k,i}(c_k^{(\mathrm{d})})}L_{k,i}(\boldsymbol{y},\alpha)\Big)+1\Big) \quad (15)\end{aligned}$$

can be found [31]. The BMI can be used to evaluate the soft decision performance of soft-input soft-output (SISO) symbol detectors in numerical simulations. For suboptimal detectors, the optimum α which maximizes the BMI can be determined in an efficient way, e.g., by the Golden section search [33]. For gradient descent optimization, we employ the metric in (15) with $\alpha=1$ as an objective function.

For the optimization of the GAP algorithm with multiple stages $S>1$, we can increase the gradient update of the backpropagation by using multiloss terms [34]. Hence, we suggest the term

$$\begin{aligned}\mathrm{BMI}_\mathrm{multi} :=\ & \log_2(M) - \frac{1}{SDK}\sum_{s=1}^{S}\sum_{i=1}^{m}\sum_{k=0}^{K-1}\sum_{(\boldsymbol{c}^{(\mathrm{d})},\boldsymbol{y}^{(\mathrm{d})})\in\mathcal{D}} \\ & \times \log_2\Big(\exp\Big(-(-1)^{b_{k,i}(c_k^{(\mathrm{d})})}L_{k,i}^{(s)}(\boldsymbol{y},\alpha)\Big)+1\Big) \quad (16)\end{aligned}$$

as the average BMI between the transmitted bits and the LLRs obtained from the APP estimates after each stage s, which we denote with $L_{k,i}^{(s)}(\boldsymbol{y},\alpha)$. Using the multiloss term in (16) for the optimization of the GAP algorithm improves learning of the earlier stages [34]. Note that we use the default loss (15) unless explicitly stated differently.

[2]Note that the BMI is often called generalized mutual information (GMI) in literature. GMI, however, is a more general concept and defines a lower bound of the mismatched capacity. For the special case of a mismatched decoder due to bit-metric decoding, the GMI is equivalent to the BMI [32].

TABLE III
CHARACTERIZATION OF LINEAR ISI REFERENCE CHANNELS

Name	Alias	L	Impulse Response $\boldsymbol{h}$
Proakis A	$\mathrm{C_A}$	10	$(0.04, -0.05, 0.07, -0.21, -0.5, 0.72, 0.36, 0.0, 0.21, 0.03, 0.07)^{\mathrm{T}}$
Proakis B	$\mathrm{C_B}$	2	$(0.407, 0.815, 0.407)^{\mathrm{T}}$
Proakis C	$\mathrm{C_C}$	4	$(0.227, 0.46, 0.688, 0.46, 0.227)^{\mathrm{T}}$

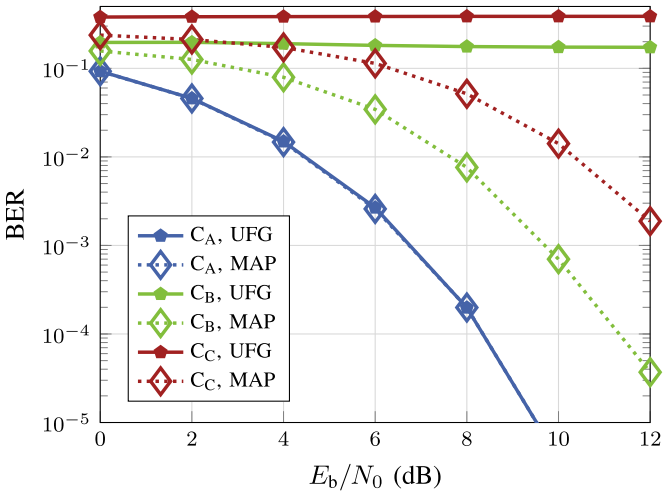

Fig. 4. BER over E_b/N_0 of the UFG algorithm for a BPSK transmission over different linear ISI channels.

The parameters S, B, N' and L_p define the general dimensionality of the model and are not part of the optimization process described in this section. These so called hyperparameters define the overall behavior and complexity of the GAP algorithm and can be either chosen by hand, or optimized in a so-called hyperparameter tuning, as elaborated in [35].

V. NUMERICAL RESULTS

We evaluate the considered symbol detectors towards their detection performance. In all simulations, the information symbols are sampled independently and uniformly from the constellation alphabet $\mathcal{M}$. If not mentioned otherwise, all iterative algorithms perform $N = 10$ iterations and the trainable parameters are optimized for $E_\mathrm{b}/N_0 = 10$ dB. The source code for the parameter optimization and evaluation of the GAP algorithm is available online [36]. The symbol-wise MAP detector, implemented by the BCJR algorithm [3], as well as a linear minimum mean squared error (LMMSE) equalizer [2, Sec. 9.4] with filter order 30 serve as references. For the latter, we transform the soft LMMSE filter output $\mathsf{c}_{k,\mathrm{LMMSE}}$ to APPs $\hat{P}(c_k|\boldsymbol{y})$ by applying a Gaussian approximation to the estimation error $\mathsf{e}_k := |\mathsf{c}_{k,\mathrm{LMMSE}} - \mathsf{c}_k| \sim \mathcal{N}\left(0, \hat{\sigma}^2_{\mathrm{LMMSE}}\right)$ and estimating the error variance $\hat{\sigma}^2_{\mathrm{LMMSE}}$ based on the hard decisions $\operatorname{argmax}_{\mathrm{m}\in\mathcal{M}} |\mathsf{c}_{k,\mathrm{LMMSE}} - \mathrm{m}|$. We consider a block length of $K = 500$ symbols and the transmission over three standard ISI channel models [2, Sec. 9.4], which are characterized in Table III.

We analyze the detection performance of the original UFG algorithm on all three reference channels. Note that with the parametrization $S = B = 1$, $\boldsymbol{P}^{(1,1)} = \boldsymbol{H}^{\mathrm{H}}$, $\mathcal{P}^{(1,1)}_\mathrm{p} = \mathcal{P}_1$ and $\mathcal{P}^{(1,1)}_\mathrm{NBP} = \mathcal{P}_1$, the GAP algorithm instantiates the UFG algorithm. Figure 4 shows the hard-decision performance of the UFG algorithm in terms of the bit error rate (BER) over E_b/N_0 for binary phase-shift keying (BPSK). The performance gap to MAP detection is highly channel specific. While the UFG algorithm operates close to optimality for the channel $\mathrm{C_A}$, its detection capabilities for the channels $\mathrm{C_B}$ and $\mathrm{C_C}$ are quite poor. Notably, the BER does not decrease for an increasing E_b/N_0.

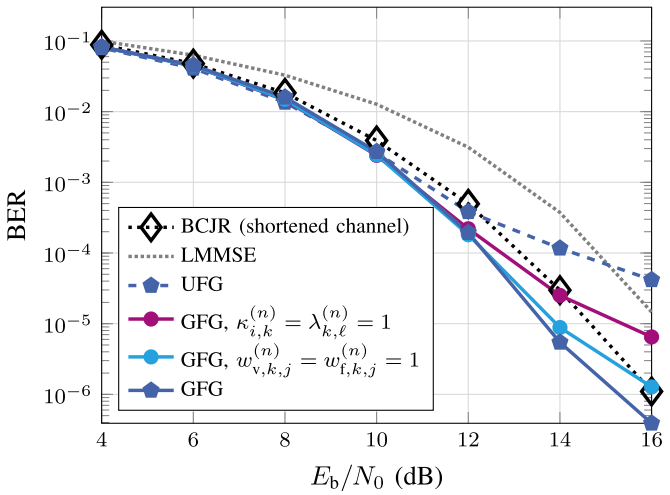

Fig. 5. Hard-decision performance of the GFG algorithm with $\boldsymbol{P} = \boldsymbol{H}^{\mathrm{H}}$ and different constraints on $\mathcal{P}_\mathrm{NBP}$ for a 16-QAM transmission over the channel $\mathrm{C_A}$. The BCJR algorithm is applied on a shortened channel with an impulse response of length 4 for complexity reasons.

A. Neural Belief Propagation and FN Enhancement

We evaluate the effects of NBP and the neural enhancement of the FNs for the factor graph-based symbol detection on channel $\mathrm{C_A}$. Since the original UFG algorithm already approaches optimum detection performance for BPSK, we consider a 16-quadrature amplitude modulation (QAM) transmission with Gray labeling.[3] Figure 5 reports the BER performance over E_b/N_0. The UFG algorithm outperforms the LMMSE equalizer in the low E_b/N_0 regime but runs into an error floor. Applying NBP and neurally enhancing the FNs of the UFG symbol detector, i.e., optimizing the parameters $\mathcal{P}_\mathrm{NBP}$ at $E_\mathrm{b}/N_0 = 14$ dB for the GFG algorithm with $\boldsymbol{P} = \boldsymbol{H}^{\mathrm{H}}$ mitigates this behavior and generalizes well over the complete E_b/N_0 range. To distinguish between the performance gain due to the weighting of the SPA messages, and the generalization of the FNs, we partly constrain the space of the gradient descent optimization over $\mathcal{P}_\mathrm{NBP}$. First, we fix $\kappa^{(n)}_{i,k} = \lambda^{(n)}_{k,\ell} = 1$, thereby disabling the FN generalization. The remaining free parameters are jointly optimized towards the BMI which yields a performance improvement. Second, we disable the message weighting by setting $w^{(n)}_{\mathrm{v},k,j} = w^{(n)}_{\mathrm{f},k,j} = 1$ and only optimize the remaining parameters within the FNs. Although the dimensionality of the optimization space is about 4 times smaller for the latter case compared to NBP, the performance gain is significantly larger. However,

[3] We expect Gray labeling to be optimal w.r.t. the BER and BMI performance of the proposed detection algorithms.

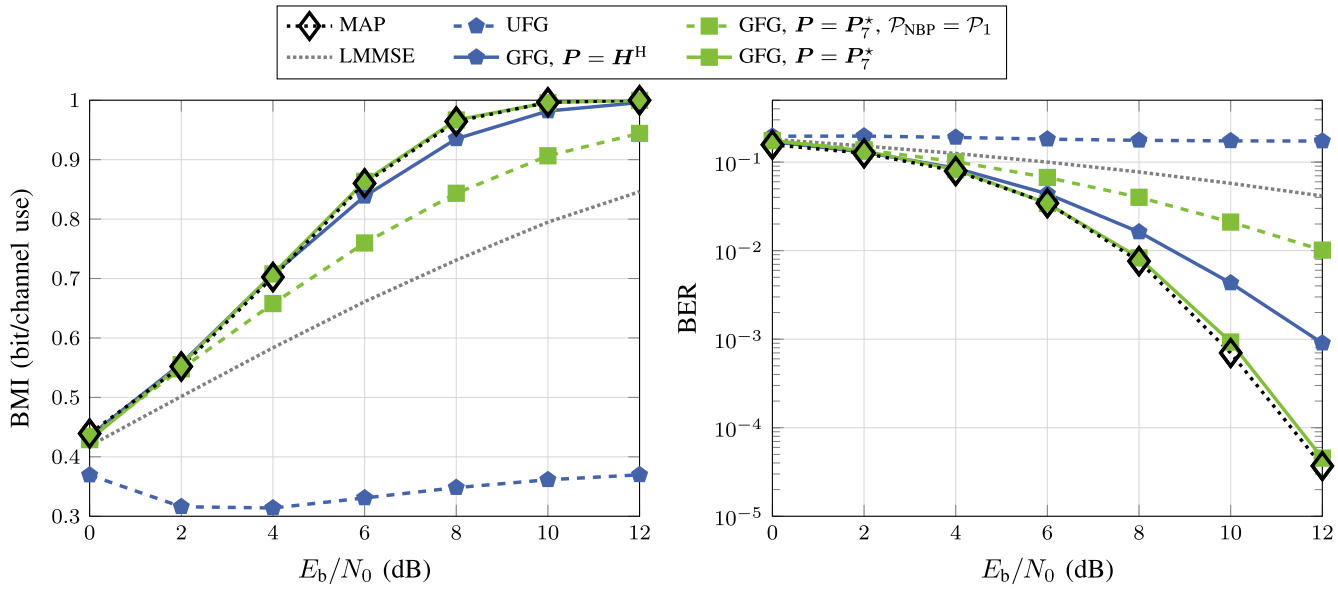

Fig. 6. BMI and BER over E_{b}/N_0 for different instances of the GFG algorithm on the channel $\mathrm{C_B}$ with BPSK signaling.

the best performance is obtained by the combination of both methods and yields a significant performance gain compared to the UFG algorithm as well as the LMMSE equalizer. Due to the constellation order $M = 16$ and a relatively large memory $L = 10$ of the channel $\mathrm{C_A}$, MAP detection becomes computationally infeasible. In order to nevertheless make a comparison of the proposed algorithm, we filter the received signal $\boldsymbol{y}$ with a channel shortening filter and then apply the BCJR algorithm on the shortened channel model. Following [7], we can derive an FIR channel shortening filter of order 25 which reduces the impulse response length of the channel from 11 ($L = 10$) to 4 ($L = 3$). The detection performance of the BCJR algorithm on the shortened model is reported in Fig. 5. It clearly outperforms the LMMSE equalizer which can be seen as MAP detection on a channel model shortened to length 1 [7]. For high $E_{\mathrm{b}}/N_0 > 12$ dB, the shortened BCJR algorithm also performs better than the conventional UFG detector, however, it cannot reach the low BERs of the neurally enhanced GFG algorithm in the considered E_{b}/N_0 range. Note that the complexity of the BCJR algorithm on the shortened channel is still more than ten times higher compared to the proposed GFG detector (complexity parameters $X = 851968$ and $X = 68825$).

B. Preprocessing

We examine the sensibility of the factor graph-based GFG algorithm to the observation model. Therefore, we consider symbol detection on the channel $\mathrm{C_B}$ with BPSK signaling in more depth. To allow a fair comparison of different observation models, we initially disable NBP and the FN generalization for the GFG algorithm by setting $\mathcal{P}_{\mathrm{NBP}} = \mathcal{P}_1$. We compare the Ungerboeck observation model with $\boldsymbol{P} = \boldsymbol{H}^{\mathrm{H}}$, which is employed by the UFG algorithm, to a generic preprocessing filter $\boldsymbol{P}_7$ of length $L_{\mathrm{p}} = 7$. The results are given in Fig. 6. Optimizing the preprocessor $\boldsymbol{P}_7^{\star}$ with respect to the BMI, the symbol detector approaches a BMI of 0.9 bit/channel use at $E_{\mathrm{b}}/N_0 = 10$ dB which is a gain of over 0.5 bit/channel use compared to the detection based on the Ungerboeck model.

Enabling NBP and the FN enhancement significantly improves the performance for both considered observation models. For the detector based on the Ungerboeck model, the BER is thereby reduced by a factor of more than 100 for $E_{\mathrm{b}}/N_0 = 12$ dB. A near-optimum symbol detector is obtained on the channel $\mathrm{C_B}$ by combining the generalized preprocessing with NBP and the FN generalization and jointly optimizing all parameters $\mathcal{P}_{\mathrm{NBP}} \cup \{\boldsymbol{P}_7\}$. Note that the GFG algorithm is a specialization of the GAP algorithm with $S = B = 1$, i.e., we do not perform a dynamic factor graph transition but only change the observation model once. An analysis of $\boldsymbol{P}_7$ as well of the optimized parameter set $\mathcal{P}_{\mathrm{NBP}}$ turned out to be not very insightful. Most of the weights approximately follow a Gaussian distribution with mean and variance altering over the iterations $n = 1, \ldots, 10$. Especially for $\kappa_{1,k}^{(n)}$, it is interesting to observe that its mean is notably smaller than 1.0 (varying from 0.5 to 0.9) for most of the iterations. This supports our hypothesis, discussed in Sec. IV-A, according to which the weights $\kappa_{1,k}^{(n)}$ can attenuate the $1/\sigma^2$ term inside the FNs of the factor graph, thereby dampening the overconfidence of the SPA messages which would otherwise lead to high approximation errors and impairments in the convergence behavior. Only in the last iteration $n = 10$, the weights $\kappa_{1,k}^{(n)}$ are amplified with an average $\kappa_{1,k}^{(10)}$ of 3.1.

To evaluate the convergence behavior of the considered algorithms, Fig. 7 illustrates the evolution of the BER over the SPA iterations n for the channel $\mathrm{C_B}$ at $E_{\mathrm{b}}/N_0 = 10$ dB. We can observe a non-convergent behavior for the UFG algorithm. The BER keeps oscillating between two points at BER $= 0.17$ and BER $= 0.2$ for $n > 4$. We can identify this lack of convergence as a major reason for the poor overall performance of the UFG algorithm. Modifying the observation model with the preprocessor $\boldsymbol{P} = \boldsymbol{P}_7^{\star}$ fixes this issue and leads to a monotone BER convergence for the GFG detector with

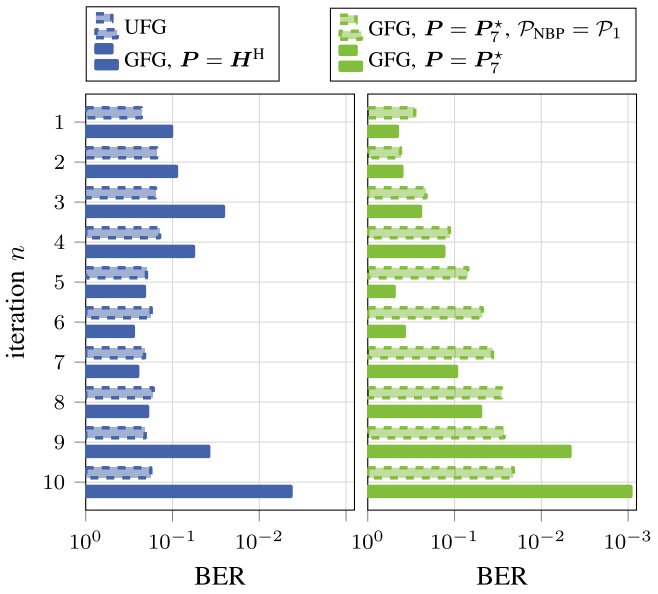

Fig. 7. BER over the iterations n of the GFG algorithm on the channel $\mathrm{C_B}$ at $E_\mathrm{b}/N_0 = 10$ dB and BPSK signaling.

$\mathcal{P}_{\mathrm{NBP}} = \mathcal{P}_1$. Applying NBP as well as the FN optimization to the GFG algorithm leads to a very interesting behavior: for both observation models, $\boldsymbol{P} = \boldsymbol{H}^{\mathrm{H}}$ and $\boldsymbol{P} = \boldsymbol{P}_7^\star$, the BER first decreases over the iterations, then it climbs again to a local maximum before it reaches its global minimum in the final iteration $n = 10$.

The impressive results of Fig. 6 raise the question of how universal the learned solutions are towards variations of the channel. Therefore, we evaluate the instances of the GFG algorithm which were specifically trained on the channel $\mathrm{C_B}$ at $E_\mathrm{b}/N_0 = 10$ dB (see Fig. 6) on the alternative channel $\tilde{\mathrm{C}}_\mathrm{B} := (0.59, 0.76, 0.28)^{\mathrm{T}}$. The channel $\tilde{\mathrm{C}}_\mathrm{B}$ was generated by adding independent and $\mathcal{N}(0, 0.1)$-distributed samples to the taps of the impulse response $\boldsymbol{h}_{\mathrm{CB}}$ of the channel $\mathrm{C_B}$ and normalizing the result with respect to the channel energy. Figure 8 compares the results to an instance of the GFG algorithm which was specifically optimized for the alternative channel $\tilde{\mathrm{C}}_\mathrm{B}$. Note that all GFG instances still have perfect channel knowledge and only their parametrization is optimized for a "mismatched" channel. We can observe that the neurally enhanced detector GFG with $\boldsymbol{P} = \boldsymbol{H}^{\mathrm{H}}$ generalizes very well. Although being trained on the channel $\mathrm{C_B}$, the performance degradation is relatively small compared to the GFG instance which was directly trained on the channel $\tilde{\mathrm{C}}_\mathrm{B}$. In contrast, the GFG algorithm with the optimized preprocessor $\boldsymbol{P} = \boldsymbol{P}_7^\star$ does not generalize at all. This makes sense, because the factor nodes are entirely based on the mismatched preprocessor $\boldsymbol{P}_7^\star$ and do not consider the true channel at all. To fix this, we impose the special structure $\boldsymbol{P} = \tilde{\boldsymbol{P}}\boldsymbol{H}^{\mathrm{H}}$ on the preprocessor and only optimize $\tilde{\boldsymbol{P}}$ for the channel $\mathrm{C_B}$ while $\boldsymbol{H}^{\mathrm{H}}$ is a matched filter based on the actual channel state information. Evaluating the optimized detector on the alternative channel $\tilde{\mathrm{C}}_\mathrm{B}$ shows that the GFG algorithm also performs well on channels which (slightly) differ from the channel for which the algorithm was optimized, under the condition that the detector has access to perfect channel state information and if the discussed structure of the preprocessor is used.

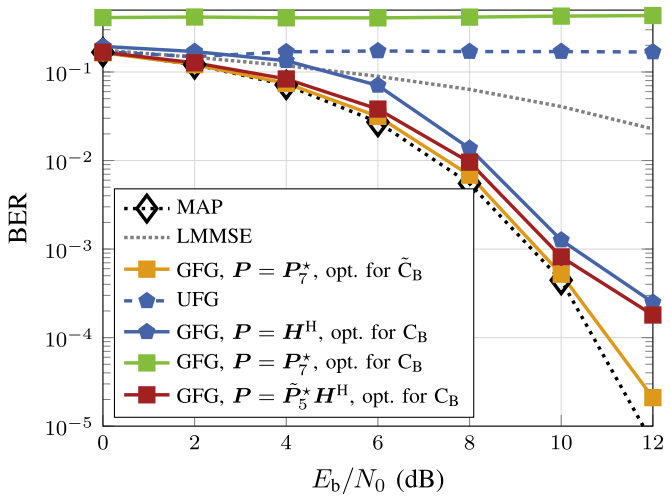

Fig. 8. BER performance of the GFG algorithm for a BPSK transmission on the channel $\tilde{\mathrm{C}}_\mathrm{B}$. The parameters of the various GFG instances were optimized for different channels.

C. Dynamic Factor Graph Transition

Figure 9 evaluates the performance of different symbol detection algorithms for the channel $\mathrm{C_C}$ and BPSK signaling. Applying an extended preprocessor of length $L_\mathrm{p} = 9$ with optimized filter taps $\boldsymbol{P} = \boldsymbol{P}_9^\star$ to the GFG algorithm can improve the detection performance compared to the UFG algorithm. However, without the application of NBP and the neural FN enhancement ($\mathcal{P}_{\mathrm{NBP}} = \mathcal{P}_1$), the factor graph-based symbol detection does not outperform the LMMSE equalizer but has an approximately constant BMI offset of about 0.13 bit/channel use. The application of NBP and the neural FN enhancement can further improve the GFG algorithm for both the Ungerboeck model $\boldsymbol{P} = \boldsymbol{H}^{\mathrm{H}}$ and the enhanced preprocessing $\boldsymbol{P} = \boldsymbol{P}_9^\star$, but a significant gap to MAP performance remains. We close this gap to optimal symbol detection by the GAP algorithm. Applying a dynamic factor graph transition with $S = 5$ stages and $B = 2$ parallel branches drastically improves the detection performance. Based on the motivation for the dynamic factor graph transition in Sec. IV-B, it might seem optimal to constantly alter the factor nodes after each iteration, i.e., to set $N' = 1$. In this case, however, we experienced that the convergence of the messages is impaired due to the (too) fast variation of the factor nodes. For the channels which are considered in this work, we found an empirical sweet spot for $N' = 4$. To evaluate on the effectiveness of the dynamic factor graph transition and the effect of parallel branches, we compare the performance of the GAP algorithm with $(S, B, N') = (5, 1, 4)$ (i.e., no parallelism $B = 1$ and $S = 5$ stages) to the performance of an alternative parametrization with only one stage $S = 1$ but $B = 5$ parallel branches. Note that both instances of the GAP algorithm employ the same number of GFG elements which leads to a comparable complexity. The performance evaluation in Fig. 9 reveals the effectiveness of the dynamic factor graph transition: the GAP algorithm with multiple serial stages shows a notably superior performance compared to the GAP algorithm with (only) parallel branches over the entire E_b/N_0 range $0 - 12$ dB considered. However,

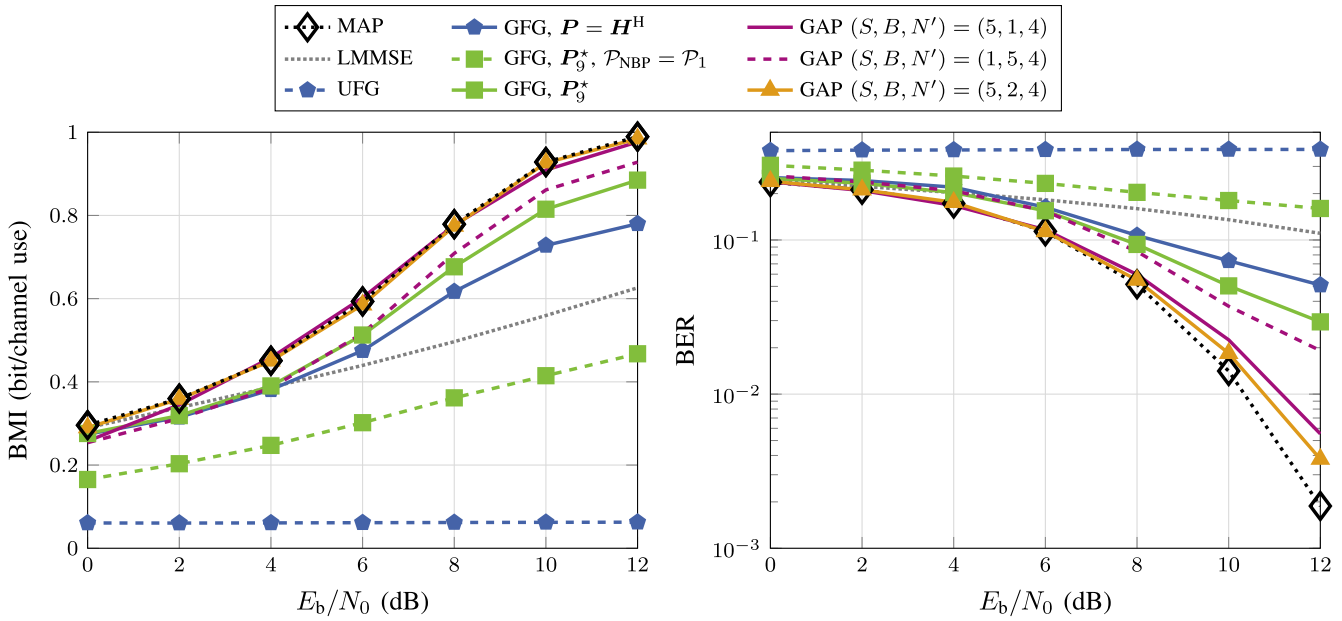

Fig. 9. Performance of different symbol detectors for a BPSK transmission over the channel $\mathrm{C_C}$. All $B \cdot S = 8$ embedded GFG units of the GAP algorithm have an individual preprocessor of length $L_{\mathrm{p}} = 9$.

the conjunction of both effects, the dynamic factor graph transition ($S = 5$) and the parallelism ($B = 2$), leads to the best overall performance. For the latter parametrization, the overall computational complexity is approximately quadrupled compared to the parametrization $(S, B, N') = (1, 1, 10)$ which instantiates a single GFG element.

We investigate the behavior of the GAP algorithm with $(S, B, N') = (5, 2, 4)$ and $L_{\mathrm{p}} = 9$ in more depth to gain insights into the effectiveness of the dynamic factor graph transition. Therefore, we analyze the performance after each stage s by approximating the BMI based on the interim APP estimates $\hat{P}_s(c_k|\boldsymbol{y})$. The results are reported in Fig. 10 (orange bars) and shows a monotonic increase of the BMI over s. To compare the contribution of both branches, we additionally determine the BMI for each branch and iteration individually, based on $\hat{P}_{s,b}(c_k|\boldsymbol{y})$. Intermediate BMI estimates for $n < N'$ are obtained by an early termination of the iterative message passing in Algorithm 1 and are denoted by $\hat{P}_{s,b}^{(n)}(c_k|\boldsymbol{y})$. Figure 10 shows the results for branch $b = 1$ on the left (red bars) and for branch $b = 2$ on the right (blue bars). The BMI evolution of the single GFG units over the iterations n is highly non-monotonic. Moreover, the two branches have a very distinct behavior. Especially after the stages $s = 2$ and $s = 5$, the GFG output $\hat{P}_{s,1}(c_k|\boldsymbol{y}) = \hat{P}_{s,1}^{(N')}(c_k|\boldsymbol{y})$ of branch $b = 1$ has a BMI close to zero. However, the combination of both branches, i.e., the combination of $\hat{P}_{s,1}(c_k|\boldsymbol{y})$ and $\hat{P}_{s,2}(c_k|\boldsymbol{y})$ to $\hat{P}_s(c_k|\boldsymbol{y})$ (orange bars), still yields an improved BMI compared to the BMI of one branch $\hat{P}_{s,2}(c_k|\boldsymbol{y})$. This non-intuitive behavior is caused by the optimization process, which is carried out in an end-to-end manner.

Finally, we evaluate the GAP algorithm with $(S, B, N') = (5, 2, 4)$ for a 16-QAM constellation over the channels $\mathrm{C_B}$ and $\mathrm{C_C}$. The simulation results for the channel $\mathrm{C_B}$ are reported in Fig. 11. Relevant for practical applications of a 16-QAM signaling is the E_{b}/N_0 range

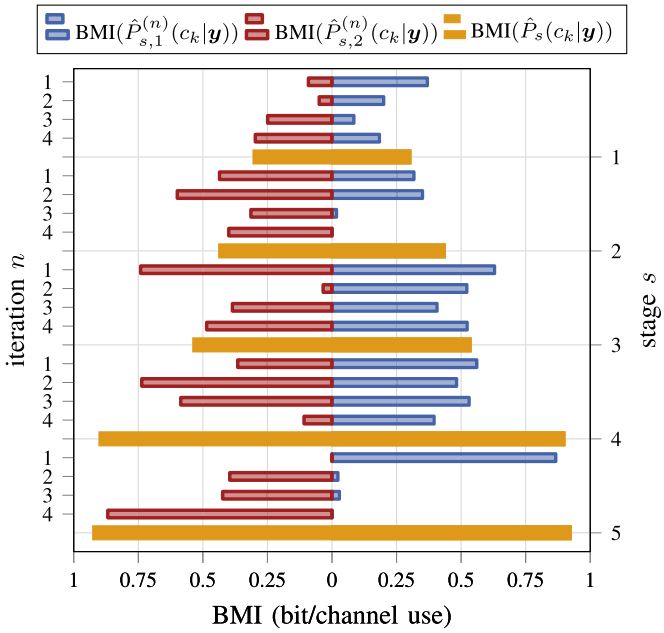

Fig. 10. Convergence behavior of the GAP algorithm with $(S, B, N') = (5, 2, 4)$ and $L_{\mathrm{p}} = 9$ for a BPSK transmission over the channel $\mathrm{C_C}$. The convergence is analyzed w.r.t. the BMI over the $S = 5$ stages (orange bars) as well as regarding the $N' = 4$ iterations within each stage, separately for both $B = 2$ branches (red and blue bars).

around $10 - 16$ dB, where the MAP detector approaches the upper BMI bound of 4 bit/channel use. Optimizing $\mathcal{P}_{\mathrm{GAP}}$ with $L_{\mathrm{p}} = 7$ for $E_{\mathrm{b}}/N_0 = 14$ dB in an end-to-end manner yields a BMI of 3.58 bit/channel use. The performance can be further improved to a BMI of 3.81 bit/channel use for $E_{\mathrm{b}}/N_0 = 14$ dB by employing the multiloss term (16) for the optimization process. The GAP detector outperforms the GFG algorithm by 0.95 bit/channel use and the LMMSE equalizer by 1.75 bit/channel use, thereby closing the gap to optimum

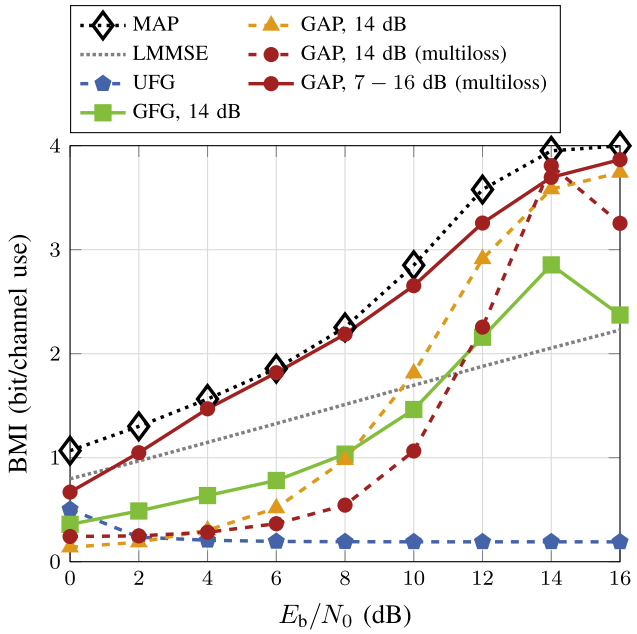

Fig. 11. BMI versus E_b/N_0 of different symbol detectors for a 16-QAM transmission over the channel C_B. The GAP algorithm is parametrized with $(S, B, N') = (5, 2, 4)$. The GFG and the GAP algorithm both apply preprocessors of length $L_p = 7$.

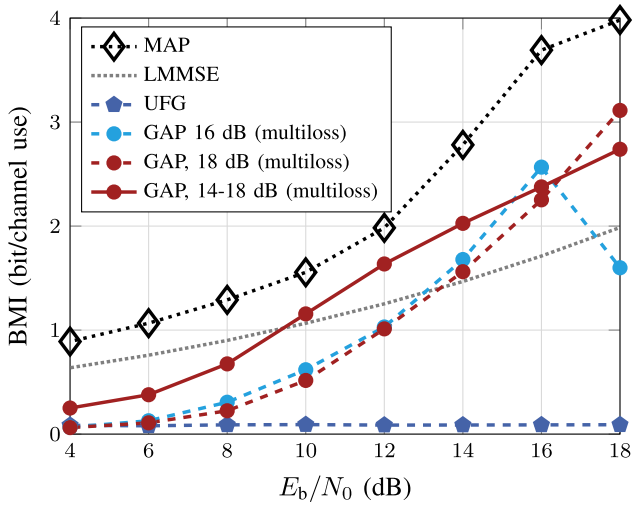

Fig. 12. BMI over E_b/N_0 of the GAP algorithm with $(S, B, N') = (5, 2, 4)$ and $L_p = 9$ for a 16-QAM transmission over the channel C_C, optimized for different ranges of E_b/N_0.

performance. However, both algorithms do not generalize well, especially for the low E_b/N_0 regime. To reduce overfitting to a specific E_b/N_0, we can vary the E_b/N_0 while optimizing $\mathcal{P}_{\text{GAP}}$. For instance, sampling the E_b/N_0 from a uniform distribution in the range $7 - 16$ dB during training consequently results in an improved average performance of the GAP algorithm in this E_b/N_0 range. Additionally, the detection performance also significantly improves for $E_b/N_0 < 7$ dB, even though this E_b/N_0 range was not sampled during the optimization. The generalized training leads to a minor BMI degradation of 0.11 bit/channel use for $E_b/N_0 = 14$ dB, compared to the optimization at a fixed $E_b/N_0 = 14$ dB. The training range of the E_b/N_0 should thus match the region of operation of the detector as accurately as possible. Further increasing the number of stages S, branches B, iterations N' or the filter order L_p has not shown any significant performance gain. The BMI performance of the GAP algorithm for the channel C_C and 16-QAM is reported in Fig. 12. We evaluate three different GAP instances, which were all optimized w.r.t. the BMI multiloss term (16), but differ in the range from which the E_b/N_0 was uniformly sampled during the training. The BMI performance of the GAP detector behaves qualitatively similar to the results on channel C_B. Trained for a specific E_b/N_0, e.g., $E_b/N_0 = 18$ dB, the GAP algorithm outperforms the LMMSE equalizer about 1.1 bit/channel use in terms of the BMI.

VI. Conclusion

We studied the application and neural enhancement of factor graph-based symbol detectors on AWGN channels with linear ISI. We proposed simple but effective generalizations of the factor graph, as well as NBP as an enhanced message passing algorithm in order to mitigate the effect of cycles in the graphs. The methods are only marginally increasing the detection complexity compared to the UFG algorithm. We further proposed the novel symbol detection algorithm GAP which comprises both NBP as well as a dynamic transition of the underlying factor graph. The algorithm delivers an attractive and highly scalable tradeoff between performance and complexity. Our methods showed a significant performance improvement of the factor graph-based symbol detector, closing the gap to optimum detection performance in various transmission scenarios. Especially for high-order constellations and static channels with large memory, the proposed GAP algorithm is a promising low-complexity alternative to the BCJR algorithm.

References

[1] L. Schmid and L. Schmalen, "Neural enhancement of factor graph-based symbol detection," in *Proc. IEEE 23rd Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, Oulu, Finland, Jul. 2022, pp. 1–4.
[2] J. Proakis and M. Salehi, *Digital Communications*, 5th ed. New York, NY, USA: McGraw-Hill, Nov. 2007.
[3] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate (corresp.)," *IEEE Trans. Inf. Theory*, vol. IT-20, no. 2, pp. 284–287, Mar. 1974.
[4] D. Fertonani, A. Barbieri, and G. Colavolpe, "Reduced-complexity BCJR algorithm for turbo equalization," *IEEE Trans. Commun.*, vol. 55, no. 12, pp. 2279–2287, Dec. 2007.
[5] V. Franz and J. B. Anderson, "Concatenated decoding with a reduced-search BCJR algorithm," *IEEE J. Sel. Areas Commun.*, vol. 16, no. 2, pp. 186–195, Feb. 1998.
[6] G. Colavolpe, G. Ferrari, and R. Raheli, "Reduced-state BCJR-type algorithms," *IEEE J. Sel. Areas Commun.*, vol. 19, no. 5, pp. 848–859, May 2001.
[7] F. Rusek and A. Prlja, "Optimal channel shortening for MIMO and ISI channels," *IEEE Trans. Wireless Commun.*, vol. 11, no. 2, pp. 810–818, Feb. 2012.
[8] F. R. Kschischang and H.-A. Loeliger, "Factor graphs and the sumproduct algorithm," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, p. 22, Feb. 2001.
[9] G. Ungerboeck, "Adaptive maximum-likelihood receiver for carrier-modulated data-transmission systems," *IEEE Trans. Commun.*, vol. COM-22, no. 5, pp. 624–636, May 1974.
[10] G. Colavolpe, D. Fertonani, and A. Piemontese, "SISO detection over linear channels with linear complexity in the number of interferers," *IEEE J. Sel. Topics Signal Process.*, vol. 5, no. 8, pp. 1475–1485, Dec. 2011.
[11] N. Shlezinger, J. Whang, Y. C. Eldar, and A. G. Dimakis, "Model-based deep learning," 2020, *arXiv:2012.08405*.

[12] N. Shlezinger, N. Farsad, Y. C. Eldar, and A. J. Goldsmith, "Data-driven factor graphs for deep symbol detection," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Los Angeles, CA, USA, Jun. 2020, pp. 1–15.
[13] V. G. Satorras and M. Welling, "Neural enhanced belief propagation on factor graphs," in *Proc. 24th Int. Conf. Artif. Intell. Statist. (AISTATS)*, San Diego, CA, USA, 2021, pp. 685–693.
[14] B. Liu, S. Li, Y. Xie, and J. Yuan, "A novel sum-product detection algorithm for faster-than-nyquist signaling: A deep learning approach," *IEEE Trans. Commun.*, vol. 69, no. 9, pp. 5975–5987, Sep. 2021.
[15] E. Nachmani, Y. Be'ery, and D. Burshtein, "Learning to decode linear codes using deep learning," in *Proc. 54th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Monticello, IL, USA, Sep. 2016, pp. 341–346.
[16] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain," in *Proc. IEEE Int. Conf. Commun.*, vol. 2, Jun. 1995, pp. 1009–1013.
[17] H.-A. Loeliger, "An introduction to factor graphs," *IEEE Signal Process. Mag.*, vol. 21, no. 1, pp. 28–41, Jan. 2004.
[18] J. Yedidia, W. T. Freeman, and Y. Weiss, "Understanding belief propagation and its generalizations," *Exploring Artif. Intell. New Millennium*, vol. 8, pp. 239–269, Jan. 2003.
[19] G. Forney, "Lower bounds on error probability in the presence of large intersymbol interference," *IEEE Trans. Commun.*, vol. COM-20, no. 1, pp. 76–77, Feb. 1972.
[20] H. L. Harney, *Bayesian Inference: Parameter Estimation and Decisions*. Heidelberg, Germany: Springer, May 2003.
[21] G. Colavolpe and G. Germi, "On the application of factor graphs and the sum-product algorithm to ISI channels," *IEEE Trans. Commun.*, vol. 53, no. 5, pp. 818–825, May 2005.
[22] G. Colavolpe and A. Barbieri, "On MAP symbol detection for ISI channels using the Ungerboeck observation model," *IEEE Commun. Lett.*, vol. 9, no. 8, pp. 720–722, Aug. 2005.
[23] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *Proc. 27th Int. Conf. Int. Conf. Mach. Learn.*, Jun. 2010, pp. 399–406.
[24] A. Krishnamurthy and A. Singh, "Sufficient statistic," in *Information Processing and Learning*. Pittsburgh, PA, USA: Carnegie Mellon Univ., 2015.
[25] T. M. Cover and J. A. Thomas, *Elements of Information Theory* (Telecommunications and Signal Processing). New York, NY, USA: Wiley, 1991.
[26] T. Hehn, J. Huber, O. Milenkovic, and S. Laendner, "Multiple-bases belief-propagation decoding of high-density cyclic codes," *IEEE Trans. Commun.*, vol. 58, no. 1, pp. 1–8, Jan. 2010.
[27] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent.*, San Diego, CA, USA, May 2015, pp. 1–15.
[28] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986.
[29] A. I. Guillén Fàbregas, A. Martinez, and G. Caire, "Bit-interleaved coded modulation," in *Found. Trends Commun. Inf. Theory*, vol. 5, nos. 1–2, pp. 1–153, 2008.
[30] L. Szczecinski and A. Alvarado, *Bit-Interleaved Coded Modulation: Fundamentals, Analysis and Design*. Chichester, U.K.: Wiley, 2015.
[31] A. Alvarado, T. Fehenberger, B. Chen, and F. M. J. Willems, "Achievable information rates for fiber optics: Applications and computations," *J. Lightw. Technol.*, vol. 36, no. 2, pp. 424–439, Jan. 15, 2018.
[32] A. Martinez, A. G. I. Fabregas, G. Caire, and F. M. J. Willems, "Bit-interleaved coded modulation revisited: A mismatched decoding perspective," *IEEE Trans. Inf. Theory*, vol. 55, no. 6, pp. 2756–2765, Jun. 2009.
[33] L. Schmalen, A. Alvarado, and R. Rios-Müller, "Performance prediction of nonbinary forward error correction in optical transmission experiments," *J. Lightw. Technol.*, vol. 35, no. 4, pp. 1015–1027, Feb. 15, 2017.
[34] E. Nachmani, E. Marciano, L. Lugosch, W. J. Gross, D. Burshtein, and Y. Beery, "Deep learning methods for improved decoding of linear codes," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 119–131, Feb. 2018.
[35] S. Raschka, "Model evaluation, model selection, and algorithm selection in machine learning," 2018, *arXiv:1811.12808*.
[36] L. Schmid. (2022). *Source Code*. [Online]. Available: https://github.com/kit-cel/gap

Luca Schmid received the B.Sc. and M.Sc. degrees in electrical engineering and information technology from the Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany, in 2018 and 2021, respectively, where he is currently pursuing the Ph.D. degree. He is also a Research and Teaching Assistant with the Communications Engineering Laboratory (CEL), KIT. His research interests include graphical models for communications and machine learning and information theory.

Laurent Schmalen (Senior Member, IEEE) received the Diploma-(Ing.) degree in electrical engineering and information technology and the Dr.-(Ing.) degree from the RWTH Aachen University of Technology, Aachen, Germany. From 2011 to 2019, he was a member of Technical Staff at the Bell Laboratories and also the Department Head of the Coding in Optical Communications Department from 2016 to 2019. From 2014 to 2019, he was a Guest Lecturer at the University of Stuttgart, Germany. He is currently a Full Professor with the Karlsruhe Institute of Technology (KIT), where he heads the Communications Engineering Laboratory (CEL). His research interests include forward error correction, modulation formats, and information theory for future optical networks. He was a recipient and a co-recipient of several awards, including the E-Plus Award for his Ph.D. thesis, the Best Paper Award of the 2010 ITG Speech Communication Conference, the 2013 Best Student Paper Award from the IEEE Signal Processing Systems Workshop, and both the 2016 and 2018 *Journal of Lightwave Technology* best paper awards. He serves as an Associate Editor for the IEEE TRANSACTIONS ON COMMUNICATIONS.