A Bidirectional List-Sequential (BI-LISS) Equalizer for Turbo Schemes

Christian Kuhn
Lehrstuhl für Nachrichtentechnik (LNT), Technische Universität München (TUM)
80290 München, Germany
Phone: +49 (89) 289 23477, Fax: +49 (89) 289 23490
Email: christian.kuhn@tum.de

***Abstract*— In the context of coded transmission over intersymbol interference channels we investigate a bidirectional list-sequential equalizer (BI-LISS) suitable for turbo schemes. Especially for channels with an unmanageable high number of states for a trellis based APP equalization this sequential algorithm shows almost optimal performance with much smaller complexity. Generally, the list-sequential (LISS) equalizer is based on a decoding technique for convolutional codes with high memory, namely sequential decoding. We introduce a BI-LISS equalizer which leads to a further reduction of the work amount and provides the possibility of parallel processing. For this purpose two independent LISS equalizers are employed working in the forward and the backward direction respectively.**

I. Introduction

For coded transmission over channels introducing intersymbol interference (ISI) receivers based on the turbo principle [1, 2] are known to achieve a performance close to theoretical limits. The equalizer and decoder components have to be soft-in/soft-out algorithms calculating a posteriori probabilities (APP) for the individual bits. In particular for channels with a long delay spread a trellis based APP equalizer suffers from a high computational load and has to be replaced with an algorithm of reduced complexity that approximates the a posteriori probabilities. Following the ideas of sequential decoding for convolutional codes with high memory [3] a tree based list-sequential (LISS) equalizer suitable for that task was introduced in [4]. Therefore, a modified Fano metric [5] is applied for exploring an equalizer tree taking into account the a priori information and an appropriate length bias for channels with memory. The tree-searching is performed using the stack algorithm [6] in order to find a subset of the most probable sequences transmitted. A soft-output is derived from those sequences contained in the stack after carrying out a soft augmentation. This technique makes fully use of the a priori information to improve the soft-output. Basically, sequential algorithms show a variable detection effort. Poor channel conditions reduce the effectivity of the stack algorithm and it stops far away from the end of the tree when the stack is filled up. Due to the soft augmentation we are able to mitigate this problem for the unidirectional LISS equalizer but further improvement can be achieved using a bidirectional list-sequential (BI-LISS) equalizer. In sequential decoding originally all implementations aim at the hard decided maximum likelihood path to be found after a maximum number of branch extentions. A bidirectional sequential decoding algorithm for that purpose is suggested in [7] searching the tree from the forward direction with a forward decoder and independently from the backward direction with a backward decoder. Decoding stops whenever either the forward or the backward decoder reaches the end of its tree. Another bidirectional approach also working with a forward and a backward decoder is presented in [8]. This decoding algorithm is successful when two appropriate partial paths of forward and backward decoder, respectively, merge in a common encoder state. Accordingly, the BI-LISS equalizer performs a forward and a backward LISS equalization each with a soft augmentation and therefore the classical problem of not finding a merging path after a finite number of branch extentions becomes obsolete in a turbo scheme.

The paper is organized as follows: In Section II the system model and basic definitions are introduced. Section III describes the BI-LISS equalizer in detail. In Section IV simulation results are presented for the BI-LISS equalizer and the unidirectional LISS equalizer. Finally, concluding remarks are given in Section V.

II. System Model

We consider the communications system depicted in Fig. 1. Binary data is encoded by an outer FEC encoder typically using a terminated or tailbited convolutional code. After interleaving subblocks $\mathbf{x}_1^N = (\mathbf{x}_1, \ldots, \mathbf{x}_n, \ldots, \mathbf{x}_N)^{\mathrm{T}}$ extending from time 1 to N with the vector elements $\mathbf{x}_n = (x_{1,n}, \ldots, x_{m,n}, \ldots, x_{M,n})$ of length M and the binary digits $x_{m,n} = \{+1, -1\}$ are formed. Each bit sequence $\mathbf{x}_n$ is mapped on a complex valued symbol $s_n(\mathbf{x}_n)$ from the 2^M-ary signal constellation, leading to the symbol vector

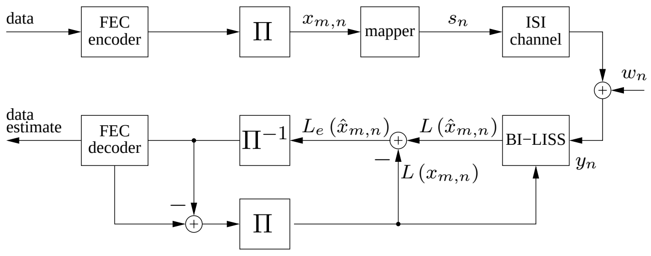


Fig. 1. Transmission system with iterative detection.

$\mathbf{s}_1^N = (s_1, \ldots, s_n, \ldots, s_N)^{\mathrm{T}}$ of length N which is appended by L terminating symbols and transmitted over the ISI channel. The receiver observes the symbol vector $\mathbf{y}_1^{N+L} = (y_1, \ldots, y_n, \ldots y_{N+L})^{\mathrm{T}}$ extending from time 1 to $N+L$ with elements

$$y_n = \sum_{l=0}^{L} h_l s_{n-l} + w_n \tag{1}$$

where $\mathbf{h} = (h_0, h_1, \ldots, h_L)^{\mathrm{T}}$ is the discrete time impulse response of the channel with memory L and unit energy, i.e. $\sum_{l=0}^{L} |h_l|^2 = 1$. The receiving process is corrupted by complex-valued additive white Gaussian noise (AWGN) with i.i.d. noise samples w_n satisfying the probability density function (pdf)

$$p(w_n) = \frac{1}{2\pi\sigma_w^2} \mathrm{e}^{-\frac{|w_n|^2}{2\sigma_w^2}} . \tag{2}$$

At the receiver we perform iterative equalization and decoding. According to the turbo principle reliability information about the code bits is interchanged between the soft-in/soft-out (SISO) components. The first one is represented by the equalizer which obtains a posteriori information about the interleaved code bits $L(\hat{x}_{m,n})$ using the received symbols $\mathbf{y}_1^{N+L}$ together with the a priori information $L(x_{m,n})$ for all individual bits. Only the extrinsic information $L_e(\hat{x}_{m,n})$ is passed to the outer SISO decoder after deinterleaving. Normally, this decoder component applies the BCJR algorithm or one of its approximations [9]. The extrinsic part of the soft decoder output is fed back and serves after interleaving as a priori input for the equalizer during the next iteration which is carried out using the same set of received data. When convergence is reached the iterative process stops and the receiver delivers the data estimates.

III. List-Sequential Turbo Equalization

A. APP soft-output

Within the considered turbo scheme the equalizer has to generate the a posteriori log-likelihood ratio (LLR)

$$\begin{aligned} L(\hat{x}_{m,n}) &= \ln \frac{P\left(x_{m,n} = +1 | \mathbf{y}_1^{N+L}\right)}{P\left(x_{m,n} = -1 | \mathbf{y}_1^{N+L}\right)} \\ &= \ln \frac{\sum_{\forall \mathbf{x}_1^N : x_{m,n}=+1} \mathrm{e}^{\ln P\left(\mathbf{x}_1^N | \mathbf{y}_1^{N+L}\right)}}{\sum_{\forall \mathbf{x}_1^N : x_{m,n}=-1} \mathrm{e}^{\ln P\left(\mathbf{x}_1^N | \mathbf{y}_1^{N+L}\right)}} \end{aligned} \tag{3}$$

for each $x_{m,n}$ which can be split up into extrinsic and a priori parts

$$\begin{aligned} L(\hat{x}_{m,n}) &= \ln \frac{p\left(\mathbf{y}_1^{N+L} | x_{m,n} = +1\right)}{p\left(\mathbf{y}_1^{N+L} | x_{m,n} = -1\right)} + \ln \frac{P(x_{m,n} = +1)}{P(x_{m,n} = -1)} \\ &= L_e(\hat{x}_{m,n}) + L(x_{m,n}). \end{aligned} \tag{4}$$

These LLRs can efficiently be calculated operating on a trellis using the finite state machine properties of the channel. For channels with high memory and/or large symbol alphabets the number of 2^{ML} channel states causes an unmanageable high computational complexity. The basic idea is now to approximate the a posteriori LLRs using the observation that the vast majority of the candidates $\mathbf{x}_1^N$ contributes a negligible amount to the total probability when evaluating the marginalization in (3).

B. APP path metric

In order to find a subset of the most probable candidates we use a modified stack algorithm exploring an equalizer tree, referred to as main stack algorithm. Evaluating (3) only for those $\mathbf{x}_1^N$ requires their corresponding APP metric values in the log-domain which can be stated as

$$\begin{aligned} \Lambda\left(\mathbf{x}_1^N\right) &= \ln P\left(\mathbf{x}_1^N | \mathbf{y}_1^{N+L}\right) \\ &= \ln p\left(\mathbf{y}_1^{N+L} | \mathbf{x}_1^N\right) + \ln P\left(\mathbf{x}_1^N\right) - \ln p\left(\mathbf{y}_1^{N+L}\right) . \end{aligned} \tag{5}$$

Note that the *channel part* $\ln p\left(\mathbf{y}_1^{N+L} | \mathbf{x}_1^N\right)$ as well as the *a priori part* $\ln P\left(\mathbf{x}_1^N\right)$ of the metric can be computed recursively in n for a given sequence $\mathbf{x}_1^N$ due to statistically independent noise samples w_n and code bits $x_{m,n}$, respectively. Based on these properties the so-called *length bias part* of the metric $\ln p\left(\mathbf{y}_1^{N+L}\right)$ can also be determined in a sequential manner. Given the initial and final channel state, the computation of (5) can be started either form the beginning of the block

$$\begin{aligned} \overrightarrow{\Lambda}\left(\mathbf{x}_1^n\right) = \overrightarrow{\Lambda}\left(\mathbf{x}_1^{n-1}\right) &+ \ln p\left(y_n | \mathbf{x}_{n-L}^n\right) + \ln P\left(\mathbf{x}_n\right) \\ &- \ln p\left(y_n | \mathbf{y}_1^{n-1}\right) \end{aligned} \tag{6}$$

in the forward direction with $\overrightarrow{\Lambda}\left(\mathbf{x}_1^1\right) = \ln p\left(y_1 | \mathbf{x}_{1-L}^1\right) + \ln P\left(\mathbf{x}_1\right) - \ln p\left(y_1\right)$ **or** from the end of the block

$$\begin{aligned} \overleftarrow{\Lambda}\left(\mathbf{x}_n^N\right) = \overleftarrow{\Lambda}\left(\mathbf{x}_{n+1}^N\right) &+ \ln p\left(y_{n+L} | \mathbf{x}_n^{n+L}\right) + \ln P\left(\mathbf{x}_n\right) \\ &- \ln p\left(y_{n+L} | \mathbf{y}_{n+L+1}^{N+L}\right) \end{aligned} \tag{7}$$

in the backward direction with $\overleftarrow{\Lambda}\left(\mathbf{x}_N^N\right) = \ln p\left(y_{N+L} | \mathbf{x}_N^{N+L}\right) + \ln P\left(\mathbf{x}_N\right) - \ln p\left(y_{N+L}\right)$. These metric calculations are carried out on a tree structure while performing branch extensions. For a full length path $\mathbf{x}_1^N$ we calculate its APP metric (5) employing either the full length forward or the full length backward metric value

$$\begin{aligned} \Lambda\left(\mathbf{x}_1^N\right) &= \overrightarrow{\Lambda}\left(\mathbf{x}_1^N\right) + \sum_{n=N+1}^{N+L} \left[\ln p\left(y_n | \mathbf{x}_{n-L}^n\right) - \ln p\left(y_n | \mathbf{y}_1^{n-1}\right)\right] \\ &= \overleftarrow{\Lambda}\left(\mathbf{x}_1^N\right) + \sum_{n=1}^{L} \left[\ln p\left(y_n | \mathbf{x}_{n-L}^n\right) - \ln p\left(y_n | \mathbf{y}_{n+1}^{N+L}\right)\right] . \end{aligned}$$

During the main stack algorithm we perform metric computations independently in the forward and backward direction for paths in the corresponding equalizer trees (see Fig. 3, solid paths). In detail the channel part of the branch metric can be expressed as

$$\ln p\left(y_n | \mathbf{x}_{n-L}^n\right) = -\frac{\left|y_n - \sum_{l=0}^{L} h_l s_{n-l}\right|^2}{2\sigma_w^2} - \ln(2\pi\sigma_w^2) \tag{8}$$

and for the a priori part of the branch metric we have

$$\ln P\left(\mathbf{x}_n\right) = \sum_{m=1}^{M}\left[x_{m,n}\frac{L(x_{m,n})}{2} - \ln\left(\mathrm{e}^{+L(x_{m,n})/2}+\mathrm{e}^{-L(x_{m,n})/2}\right)\right]. \quad (9)$$

Finally, the bias parts of the forward and backward branch metric can be obtained from

$$\ln p\left(y_n|\mathbf{y}_1^{n-1}\right) = \ln p\left(\mathbf{y}_1^{n}\right) - \ln p\left(\mathbf{y}_1^{n-1}\right) \quad (10)$$

and

$$\ln p\left(y_{n+L}|\mathbf{y}_{n+L+1}^{N+L}\right) = \ln p\left(\mathbf{y}_{n+L}^{N+L}\right) - \ln p\left(\mathbf{y}_{n+L+1}^{N+L}\right). \quad (11)$$

Although the bias term has no influence on the soft-output calculation (3), it is absolutely necessary to compare the metric values of the different length paths which occur while performing the main stack algorithm. Thus the length bias is used to speed up the tree search. Before presenting a procedure to determine the bias term we will give a detailed explanation of the main stack algorithm.

C. Main stack algorithm with soft augmentation

Depending on the equalization direction the two independent main stack algorithms work on the main stack $\overset{\rightharpoonup}{\mathcal{S}}$ or $\overset{\leftharpoonup}{\mathcal{S}}$, respectively. The TOP stack entry of each main stack corresponds to the path with highest metric which has not yet reached full length. A flow chart is depicted in Fig. 2 and in the following we will only point out the differences to the stack algorithm. Originally, the stack algorithm works only in

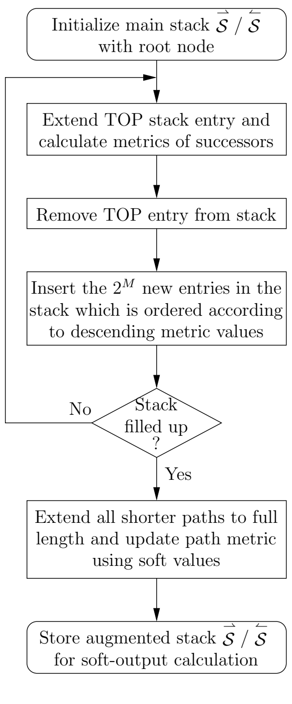

Initialize main stack $\overset{\rightharpoonup}{\mathcal{S}}$ / $\overset{\leftharpoonup}{\mathcal{S}}$ with root node
Extend TOP stack entry and calculate metrics of successors
Remove TOP entry from stack
Insert the 2^M new entries in the stack which is ordered according to descending metric values
Stack filled up ?
No
Yes
Extend all shorter paths to full length and update path metric using soft values
Store augmented stack $\overset{\rightharpoonup}{\mathcal{S}}$ / $\overset{\leftharpoonup}{\mathcal{S}}$ for soft-output calculation

Fig. 2. Flow chart for the stack algorithm with soft augmentation.

the forward direction and stops when the path with the highest metric has reached full length N. In contrast to that the main stack algorithm stops when the corresponding stack is filled up, i.e. the number of paths in the stack reached a maximum value of $|\overset{\rightharpoonup}{\mathcal{S}}|_{\max}/|\overset{\leftharpoonup}{\mathcal{S}}|_{\max}$, to use as many full length sequences as possible for the soft-output processing in (3) under a given memory constraint. When the stack is filled up with different length paths, an additional improvement of the soft-output is achieved by carrying out a full length augmentation without increasing the stack size. More precisely, we use the a priori information $L(x_{m,n})$ by means of soft bits

$$\bar{x}_{m,n} = \mathrm{E}\{x_{m,n}\} = \tanh(L(x_{m,n})/2) \quad (12)$$

as well as soft symbols

$$\bar{s}_n = \mathrm{E}\{s_n\} = \sum_{\forall(b_1,\dots,b_M)}\left[s(b_1,\dots,b_M)\prod_{m=1}^{M}\frac{1+b_m\bar{x}_{m,n}}{2}\right], \quad (13)$$

$b_m \in \{\pm1\}$, for the augmentation and the required metric calculations. In Fig. 3 an example for the forward and backward tree is presented. Solid lines belong to hard decisions and dashed lines represent the soft augmentation. These augmented

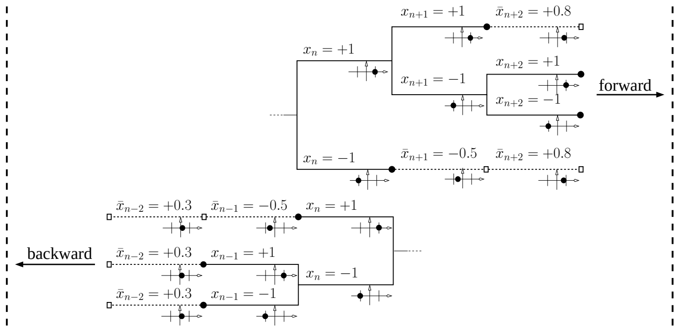


Fig. 3. Equalizer subtrees for BI-LISS applied to BPSK modulation.

paths $\breve{\mathbf{x}}_1^N$ generally contain soft values $\bar{x}_{m,n}$ and hard decided values $x_{m,n}$ as shown in Tab. I.

k	$\breve{x}_1$	…	$\breve{x}_{n-1}$	$\breve{x}_n$	$\breve{x}_{n+1}$	…	$\breve{x}_N$	$\Lambda\left(\breve{\mathbf{x}}_1^N\right)$
1	$+$	…	$-$	$+$	$-$	…	$+$	-0.9
2	$+$	…	$-$	$+$	$+$	…	$+0.1$	-1.2
3	$+$	…	$+$	$+$	$-$	…	$-$	-1.8
⋮	⋮		⋮	⋮	⋮		⋮	⋮
$\lvert\overset{\rightharpoonup}{\mathcal{S}}\rvert_{\max}$	$-$	…	$+$	$-$	-0.5	…	$+0.1$	-18.2
1	$+0.7$	…	$-$	$+$	$-$	…	$+$	-1.2
2	$+0.7$	…	$+$	$-$	$-$	…	$+$	-1.6
3	$+0.7$	…	$-$	$+$	$-$	…	$-$	-2.2
⋮	⋮		⋮	⋮	⋮		⋮	⋮
$\lvert\overset{\leftharpoonup}{\mathcal{S}}\rvert_{\max}$	$+0.7$	…	-0.5	$-$	$+$	…	$-$	-22.1
$L(\hat{x}_n)$	$+3.1$	…	-0.6	$+2.5$	-0.9	…	$+2.9$	—

TABLE I
EXAMPLE FOR THE AUGMENTED STACKS OF A BI-LISS EQUALIZER FOR BPSK TRANSMISSION WITH STACK SIZES $|\overset{\rightharpoonup}{\mathcal{S}}|_{\max}$ AND $|\overset{\leftharpoonup}{\mathcal{S}}|_{\max}$.

D. Auxiliary stack algorithm for bias estimation

The length bias parts of the branch metrics (10) and (11) are essential for the main stack algorithm to be successful. Evaluating

$$\ln p\left(\mathbf{y}_1^n\right) = \ln \sum_{\forall \mathbf{x}_1^n} \mathrm{e}^{\ln p(\mathbf{y}_1^n|\mathbf{x}_1^n)+\ln P(\mathbf{x}_1^n)} \tag{14}$$

yields the exact value of the forward partial bias term and

$$\ln p\left(\mathbf{y}_{n+L}^{N+L}\right) = \ln \sum_{\forall \mathbf{x}_n^N} \mathrm{e}^{\ln p\left(\mathbf{y}_{n+L}^{N+L}|\mathbf{x}_n^N\right)+\ln P\left(\mathbf{x}_n^N\right)} \tag{15}$$

the exact value of the backward partial bias term. Unfortunately, the calculation of (14) and (15) is far too complex, but the corresponding sums are dominated only by a small subset of sequences of a particular length. These sequences maximizing the forward partial auxiliary metrics $\overrightarrow{\Lambda}_x\left(\mathbf{x}_1^n\right) = \ln p\left(\mathbf{y}_1^n|\mathbf{x}_1^n\right) + \ln P\left(\mathbf{x}_1^n\right)$ or the backward partial auxiliary metrics $\overleftarrow{\Lambda}_x\left(\mathbf{x}_n^N\right) = \ln p\left(\mathbf{y}_{n+L}^{N+L}|\mathbf{x}_n^N\right) + \ln P\left(\mathbf{x}_n^N\right)$, respectively, can also be found by exploring the equalizer trees. For this we use a slightly modified M-algorithm, referred to as auxiliary stack algorithm. In the forward direction this algorithm works on an auxiliary stack $\overrightarrow{\mathcal{S}}_x$ with stack size $|\overrightarrow{\mathcal{S}}_x|_{\max} = 2^{M\overrightarrow{N}_x}$ and recursively computes partial auxiliary metric values

$$\overrightarrow{\Lambda}_x\left(\mathbf{x}_1^n\right) = \overrightarrow{\Lambda}_x\left(\mathbf{x}_1^{n-1}\right) + \ln p\left(y_n|\mathbf{x}_{n-L}^n\right) + \ln P\left(\mathbf{x}_n\right) \tag{16}$$

using the channel part (8) as well as the a priori part (9) of the branch metric, starting from $\overrightarrow{\Lambda}_x\left(\mathbf{x}_1^1\right) = \ln p\left(y_1|\mathbf{x}_{1-L}^1\right) + \ln P\left(\mathbf{x}_1\right)$. Consequently, the forward partial bias term can be estimated by

$$\ln p\left(\mathbf{y}_1^n\right) \approx \ln \overrightarrow{\beta}_n = \ln \sum_{\forall \mathbf{x}_1^n \in \overrightarrow{\mathcal{S}}_x} \mathrm{e}^{\overrightarrow{\Lambda}_x\left(\mathbf{x}_1^n\right)}. \tag{17}$$

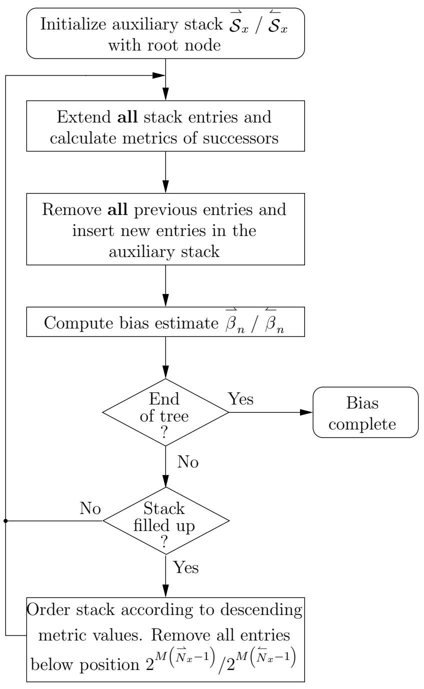
A flow chart for the auxiliary stack algorithm is depicted in Fig. 4. Similarly, we have for the backward direction an auxiliary stack $\overleftarrow{\mathcal{S}}_x$ with size $|\overleftarrow{\mathcal{S}}_x|_{\max} = 2^{M\overleftarrow{N}_x}$ and the partial auxiliary metric

$$\overleftarrow{\Lambda}_x\left(\mathbf{x}_n^N\right) = \overleftarrow{\Lambda}_x\left(\mathbf{x}_{n+1}^N\right) + \ln p\left(y_{n+L}|\mathbf{x}_n^{n+L}\right) + \ln P\left(\mathbf{x}_n\right) \tag{18}$$

starting from $\overleftarrow{\Lambda}_x\left(\mathbf{x}_N^N\right) = \ln p\left(y_{N+L}|\mathbf{x}_N^{N+L}\right) + \ln P\left(\mathbf{x}_N\right)$. An estimation of the backward partial bias is obtained by

$$\ln p\left(\mathbf{y}_{n+L}^{N+L}\right) \approx \ln \overleftarrow{\beta}_n = \ln \sum_{\forall \mathbf{x}_n^N \in \overleftarrow{\mathcal{S}}_x} \mathrm{e}^{\overleftarrow{\Lambda}_x\left(\mathbf{x}_n^N\right)}. \tag{19}$$

Contrary to the main stacks, the auxiliary stacks contain only sequences of a particular length. Moreover, to reach the end of the trees and to guarantee limited stack sizes we have to discard paths while performing the auxiliary stack algorithms for the tree depth $n > \overrightarrow{N}_x$ in the forward direction and for $n \leq N - \overleftarrow{N}_x$ in the backward direction.



Initialize auxiliary stack $\overrightarrow{\mathcal{S}}_x$ / $\overleftarrow{\mathcal{S}}_x$ with root node
Extend **all** stack entries and calculate metrics of successors
Remove **all** previous entries and insert new entries in the auxiliary stack
Compute bias estimate $\overrightarrow{\beta}_n$ / $\overleftarrow{\beta}_n$
End of tree ?
Yes
Bias complete
No
Stack filled up ?
No
Yes
Order stack according to descending metric values. Remove all entries below position $2^{M\left(\overrightarrow{N}_x-1\right)}/2^{M\left(\overleftarrow{N}_x-1\right)}$

Fig. 4. Flow chart for the auxiliary stack algorithm.

E. Soft-output processing

We now have available a set of augmented paths $\breve{\mathbf{x}}_1^N$ in the forward main stack $\overrightarrow{\mathcal{S}}$ and in the backward main stack $\overleftarrow{\mathcal{S}}$. Due to the soft augmented part we have to modify (3) to control the metric influence on numerator and denominator. Therfore, we introduce the weighting factors $(1 \pm \breve{x}_{m,n})/2$ which leads to

$$L(\hat{x}_{m,n}) \approx \ln \frac{\sum\limits_{\forall \breve{\mathbf{x}} \in \overrightarrow{\mathcal{S}}} \frac{1+\breve{x}_{m,n}}{2} \cdot \mathrm{e}^{\Lambda_x\left(\breve{\mathbf{x}}_1^N\right)} + \sum\limits_{\forall \breve{\mathbf{x}} \in \overleftarrow{\mathcal{S}}} \frac{1+\breve{x}_{m,n}}{2} \cdot \mathrm{e}^{\Lambda_x\left(\breve{\mathbf{x}}_1^N\right)}}{\sum\limits_{\forall \breve{\mathbf{x}} \in \overrightarrow{\mathcal{S}}} \frac{1-\breve{x}_{m,n}}{2} \cdot \mathrm{e}^{\Lambda_x\left(\breve{\mathbf{x}}_1^N\right)} + \sum\limits_{\forall \breve{\mathbf{x}} \in \overleftarrow{\mathcal{S}}} \frac{1-\breve{x}_{m,n}}{2} \cdot \mathrm{e}^{\Lambda_x\left(\breve{\mathbf{x}}_1^N\right)}}. \tag{20}$$

For the hard decided part of the tree we have almost the original formula, but for the soft decided part we weight the metric values with the corresponding bit probabilities

$$P(x_{m,n} = \pm 1) = \frac{1 \pm \bar{x}_{m,n}}{2}. \tag{21}$$

An important difference according to (3) is that we only have the estimated bias parts $\ln \overrightarrow{\beta}_{N+L}$ and $\ln \overleftarrow{\beta}_{-L}$, which are in general different from each other, instead of the exact value $\ln p\left(\mathbf{y}_1^{N+L}\right)$ which is valid for both directions. In order to prevent BI-LISS to privilege paths of the augmented stack with the smaller estimated full length bias, we have to use the metrics without the estimated bias terms

$$\begin{aligned} \Lambda_x\left(\breve{\mathbf{x}}_1^N\right)\Big|_{\breve{\mathbf{x}}_1^N \in \overrightarrow{\mathcal{S}}} &= \Lambda\left(\breve{\mathbf{x}}_1^N\right) + \ln \overrightarrow{\beta}_{N+L}\,, \\ \Lambda_x\left(\breve{\mathbf{x}}_1^N\right)\Big|_{\breve{\mathbf{x}}_1^N \in \overleftarrow{\mathcal{S}}} &= \Lambda\left(\breve{\mathbf{x}}_1^N\right) + \ln \overleftarrow{\beta}_{-L} \end{aligned} \tag{22}$$

for the evaluation of (20). For that reason it is only necessary to estimate the partial bias up to $\ln \overrightarrow{\beta}_N$ in the forward direction and up to $\ln \overleftarrow{\beta}_1$ in the backward direction. In conclusion the structure of BI-LISS is presented in Fig. 5.

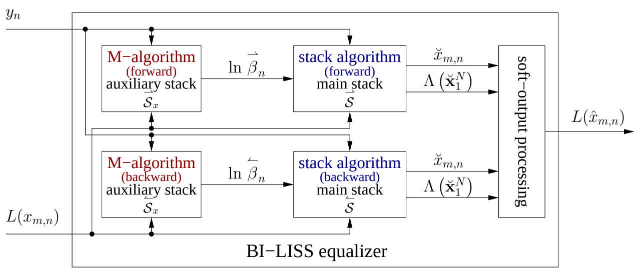

y_n
M−algorithm (forward) auxiliary stack $\overrightarrow{\mathcal{S}}_x$
$\ln \overrightarrow{\beta}_n$
stack algorithm (forward) main stack $\overrightarrow{\mathcal{S}}$
$\check{x}_{m,n}$
$\Lambda(\check{\mathbf{x}}_1^N)$
soft−output processing
$L(\hat{x}_{m,n})$
M−algorithm (backward) auxiliary stack $\overleftarrow{\mathcal{S}}_x$
$\ln \overleftarrow{\beta}_n$
stack algorithm (backward) main stack $\overleftarrow{\mathcal{S}}$
$\check{x}_{m,n}$
$\Lambda(\check{\mathbf{x}}_1^N)$
$L(x_{m,n})$
BI−LISS equalizer

Fig. 5. Structure of the BI-LISS equalizer.

IV. Simulation Results

The ionosphere is a very challenging physical channel which suffers from a large delay spread and is typically modeled by two propagation paths with equal power. Therefore, we consider the 16-tap channel $\mathbf{h} = (1/\sqrt{2}, 0, 0, \ldots 0, 0, 1/\sqrt{2})^{\mathrm{T}}$ for binary transmission [10]. Blocks of $N = 128$ symbols are arranged in an interleaver frame of size 8192 bits. The outer rate 1/2 convolutional code is terminated and defined by the generator $g = [1 + D + D^3 + D^4, 1 + D^3 + D^4]$. In Fig. 6 the BER after decoding for a turbo receiver employing BI-LISS with $|\overrightarrow{\mathcal{S}}|_{\max} = |\overleftarrow{\mathcal{S}}|_{\max} = 4096$ and $|\overrightarrow{\mathcal{S}}_x|_{\max} = |\overleftarrow{\mathcal{S}}_x|_{\max} = 128$ is depicted. Note that the full tree has 2^{128} paths. The system performance is compared to coded transmission over an AWGN channel with APP decoding. We reach this performance bound in iteration 12 at a BER

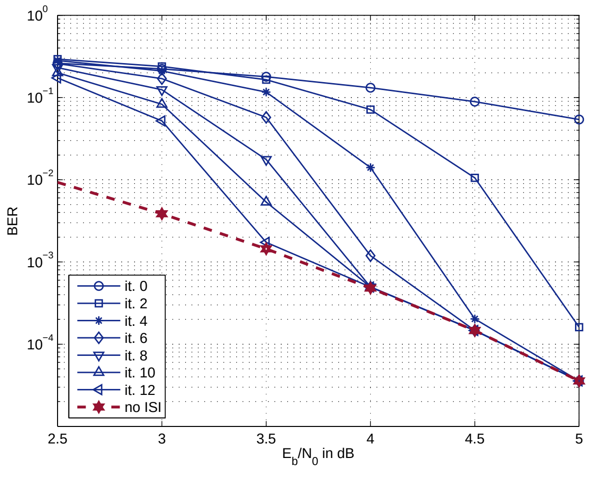


Fig. 6. BER performance after the decoder for $N = 128$, an inner BI-LISS equalizer with $|\overrightarrow{\mathcal{S}}|_{\max} = |\overleftarrow{\mathcal{S}}|_{\max} = 4096$, $|\overrightarrow{\mathcal{S}}_x|_{\max} = |\overleftarrow{\mathcal{S}}_x|_{\max} = 128$ and an outer BCJR decoder for the rate $1/2$, memory 4 convolutional code.

of 10^{-3}. In Fig. 7 a comparison of different LISS based equalizers after 12 iterations equalization and decoding is presented. For each equalizer we have fairly small auxiliary stack sizes of $|\overrightarrow{\mathcal{S}}_x|_{\max} + |\overleftarrow{\mathcal{S}}_x|_{\max} = 256$. Compared to the unidirectional LISS equalizer with $|\overrightarrow{\mathcal{S}}|_{\max} = 8192$, a turbo

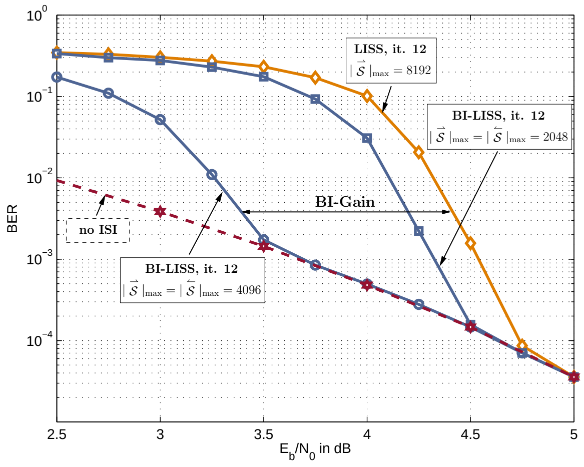


Fig. 7. BER performance after 12 iterations equalization and decoding for $N = 128$, various inner LISS based equalizers with $|\overrightarrow{\mathcal{S}}_x|_{\max} + |\overleftarrow{\mathcal{S}}_x|_{\max} = 256$ and an outer BCJR decoder for the rate $1/2$, memory 4 conv. code.

receiver employing BI-LISS with equal computational complexity, i.e. $|\overrightarrow{\mathcal{S}}|_{\max} = |\overleftarrow{\mathcal{S}}|_{\max} = 4096$, reaches the performance bound at a $1.25\mathrm{dB}$ lower E_b/N_0. Moreover, we reduced the main stack sizes by factors of two for the BI-LISS equalizer. Even in that case, BI-LISS shows a better performance.

V. Conclusion

We have introduced a bidirectional list-sequential (BI-LISS) equalizer which overcomes the problems of former bidirectional sequential decoding algorithms. Furthermore, the independent components of BI-LISS provide the possibility of parallel processing. Especially in the low E_b/N_0-range BI-LISS shows a better performance than its unidirectional complement LISS.

References

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near shannon limit error-correcting coding and decoding: Turbo codes (1)," in *Proc. of the IEEE Int. Conf. on Communications (ICC)*, vol. 2, pp. 1064–1070, May 1993.

[2] J. Hagenauer, "The turbo principle: Tutorial introduction and state of the art," in *Proc. of the Int. Symposium on Turbo Codes & Related Topics*, pp. 1–11, Sept. 1997.

[3] J. M. Wozencraft and B. Reiffen, *Sequential Decoding*. Cambridge, Mass.: MIT Press, 1961.

[4] J. Hagenauer and C. Kuhn, "Turbo equalization for channels with high memory using a list-sequential (LISS) equalizer," in *Proc. of the Int. Symposium on Turbo Codes & Related Topics*, pp. 9–13, Sept. 2003.

[5] R. Fano, "A heuristic discussion of probabilistic decoding," *IEEE Transactions on Information Theory*, vol. 9, pp. 64–73, Apr. 1963.

[6] S. Lin and D. J. Costello, *Error Control Coding: Fundamentals and Applications*. Prentice-Hall, Englewood Cliffs, 2nd ed., 2004.

[7] G. D. Forney, "Final report on a coding system design for advanced solar missions," in *Rep. NAS2-3637, Contract NASA CR73167, NASA Ames Res. Ctr.*, 1967.

[8] S. Kallel and K. Li, "Bidirectional sequential decoding," *IEEE Transactions on Information Theory*, vol. 43, pp. 1319–1326, July 1997.

[9] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Transactions on Information Theory*, vol. 42, pp. 429–445, Mar. 1996.

[10] R. Otnes and M. Tüchler, "Block SISO linear equalizers for turbo equalization in serial-tone HF modems," in *Proceedings of the Norwegian Signal Processing Symposium (NORSIG)*, pp. 93–98, Oct. 2001.