EURASIP Journal on Advances in Signal Processing
a SpringerOpen Journal

RESEARCH Open Access

Reduced complexity turbo equalization using a dynamic Bayesian network

Hermanus C Myburgh[1*], Jan C Olivier[2] and Augustinus J van Zyl[3]

Abstract

It is proposed that a dynamic Bayesian network (DBN) is used to perform turbo equalization in a system transmitting information over a Rayleigh fading multipath channel. The DBN turbo equalizer (DBN-TE) is modeled on a single directed acyclic graph by relaxing the Markov assumption and allowing weak connections to past and future states. Its complexity is exponential in encoder constraint length and approximately linear in the channel memory length. Results show that the performance of the DBN-TE closely matches that of a traditional turbo equalizer that uses a maximum *a posteriori* equalizer and decoder pair. The DBN-TE achieves full convergence and near-optimal performance after small number of iterations.

Keywords: Turbo equalizer, Dynamic Bayesian network, Rayleigh fading, Low complexity

1 Introduction

Turbo equalization has its origin in the *Turbo Principle*, first proposed in [1] where it was applied to the iterative decoding of concatenated convolutional codes. The concept of Turbo Coding was subsequently applied to equalization in [2,3] by viewing the frequency selective channel as an inner code and replacing one of the convolutional maximum *a posteriori* (MAP) decoders with a MAP equalizer. The MAP equalizer and the MAP decoder iteratively exchange extrinsic information. By iterating the system a number of times, the bit error rate (BER) performance can be improved significantly, but at the cost of additional computational complexity, especially if the frequency selective channel has long memory.

Turbo equalization becomes exceedingly complex in terms of the number of computations, due to the high computational complexity of the MAP equalizer and MAP decoder most often used in a turbo equalizer. The complexity of the MAP equalizer is linear in the data block length N, but grows exponentially with an increase in channel memory. Its complexity is therefore $O(NM^{L-1})$, where L is the channel impulse response (IR) length, and M is the modulation alphabet size. Similarly, the complexity of the MAP decoder is linear in the data block length and exponential in the encoder constraint length K.

It is however suggested in [4], and discussed in length in [5], that an MMSE equalizer can be modified for use in a turbo equalizer, to take advantage of prior information on the symbols to be estimated. By replacing the optimal MAP equalizer with a suboptimal, low complexity MMSE equalizer, low complexity turbo equalization is achieved, while still achieving matched filter bound performance using static channels of length five [5]. It was also shown in [5] how a decision feedback equalizer can be used in a turbo equalizer configuration. Also, a soft-feedback equalizer (SFE) was proposed in [6] with performance superior to that proposed in [5]. The author of [6] expanded upon ideas proposed in [5], where hard decisions on the equalizer output are fed back by combining prior information with soft decisions [6]. The performance of the SFE turbo equalizer was evaluated for a magnetic recording channel (9 taps), a microwave channel (44 taps), and a power-line channel (58 taps), outperforming the low complexity turbo equalizers proposed in [5], while doing so at reduced complexity. Still, neither of the low complexity turbo equalizers proposed in [4-6] can achieve optimal or even near-optimal results, due to the suboptimality of their constituent soft-input soft-output equalizers.

An interleaver is used as standard in a turbo equalizer not only to mitigate the effect of burst errors by

*Correspondence: herman.myburgh@up.ac.za
[1] Department of Electrical, Electronic and Computer Engineering, University of Pretoria, Pretoria 0002, South Africa
Full list of author information is available at the end of the article

randomizing the occurrence of bit errors in a transmitted data block, but also to aid in the dispersion of the positive feedback effect, which is due to the fact that the MAP algorithm used for equalization and decoding produces outputs that are locally highly correlated [4]. When a random interleaver is used, the Markov assumption, stating that the current state is only dependent on a finite history of previous states, is violated since the interleaver randomizes the encoded data according to some predetermined random permutation. The Markov assumption therefore fails and the turbo equalizer can no longer be modeled as a directed acyclic graph (DAG) to form a cycle-free decision tree. As a result, much attention has been given to approximate inference using belief propagation on graphs with cycles. The junction tree algorithm is used to combine nodes into super-nodes until the graph has no cycles [7], with an exponential growth in complexity as nodes are combined. Apart from the very high computational complexity of this approach, it has been shown in [8-10] that exact inference is not guaranteed on graphs with cycles.

In this article, a low complexity near-optimal dynamic Bayesian network turbo equalizer (DBN-TE) is proposed. The DBN-TE is modeled as a DAG, while relaxing the Markov assumption. The DBN-TE model ensures that there is always one dominant connection between a given hidden state and its corresponding observation, while there may be many weak connections to past and future hidden states. The computational complexity of the DBN-TE is exponential in decoder constraint length and approximately linear in the channel memory length. Additional complexity is due to the channel memory, but is only approximately linear since it does not increase the size of the state space, but merely increases the summation terms in the sensor model. Results show that the performance of the DBN-TE closely matches that of a conventional turbo equalizer in Rayleigh fading channels, achieving full convergence after only a small number of iterations.

This article is structured as follows. Section 2 provides a brief overview of conventional turbo equalization. Section 3 presents a discussion on the implications of modeling a turbo equalizer as a DAG and quasi-DAG, while a theoretical discussion on the iterative convergence of a quasi-DAG is discussed in Section 4. The DBN-TE formulation is discussed in Section 5 and a complexity comparison between the DBN-TE and the conventional turbo equalizer is shown in Section 6. Section 7 presents simulation results and conclusions are drawn in Section 8.

2 Turbo equalization

A turbo decoder uses two MAP decoders to iteratively decode convolutional coded concatenated codes. Like the MAP equalizer, the MAP decoder produces posterior probabilistic information on the source symbols. The output of each decoder is therefore used to produce prior probabilistic information about the input symbols of the other decoder, thus allowing this scheme to exploit the inherent structure of the code to correct errors with each iteration [11], achieving near Shannon limit performance in AWGN channels [1].

Since the communication channel can be viewed as a non-binary convolutional encoder, the channel can be viewed as an inner-code while a convolutional encoder is used as an outer-code in much the same way as in turbo coding [3], so that the turbo principle can be applied to channel equalization. As such, one of the MAP decoders in the turbo decoder is substituted with a MAP equalizer to mitigate the effect of the channel on the transmitted symbols (to "decode" the ISI-corrupted received symbols) [3]. The output of the MAP equalizer is used to produce prior probabilistic information on the encoded symbols, which is exploited by the MAP decoder. In turn, the output of the MAP decoder is used to produce prior probabilistic information on the unequalized received symbols, which is again exploited by the MAP equalizer. By iterating this system a number of times, the performance of the system can be enhanced greatly [2-5].

Figure 1 shows the structure of the turbo equalizer. The MAP equalizer takes as input the ISI-corrupted received symbols $\mathbf{r}$ and the extrinsic information $L_e^D(\hat{\mathbf{s}})$ and produces a sequence of posterior transmitted symbol log-likelihood ratio (LLR) estimates $L^E(\hat{\mathbf{s}})$ (note that $L_e^D(\hat{\mathbf{s}})$ is zero during the first iteration). Extrinsic information $L_e^E(\hat{\mathbf{s}})$ is determined by

$$L_e^E\left(\hat{\mathbf{s}}\right) = L^E\left(\hat{\mathbf{s}}\right) - L_e^D\left(\hat{\mathbf{s}}\right), \tag{1}$$

which is deinterleaved to produce $L_e^E(\hat{\mathbf{s}}')$, which is used as input to the MAP decoder to produce a sequence of posterior coded symbol LLR estimates $L^D(\hat{\mathbf{s}}')$. $L^D(\hat{\mathbf{s}}')$ is used together with $L_e^E(\hat{\mathbf{s}}')$ to determine the extrinsic information

$$L_e^D\left(\hat{\mathbf{s}}'\right) = L^D\left(\hat{\mathbf{s}}'\right) - L_e^E\left(\hat{\mathbf{s}}'\right), \tag{2}$$

$L_e^D(\hat{\mathbf{s}}')$ is interleaved to produce $L_e^D(\hat{\mathbf{s}})$. $L_e^D(\hat{\mathbf{s}})$ is used together with the received symbols $\mathbf{r}$ in the MAP equalizer, with $L_e^D(\hat{\mathbf{s}})$ serving to provide prior information on the received symbols. The equalizer again produces posterior information $L^E(\hat{\mathbf{s}})$ of the interleaved coded symbols. This process continues until the outputs of the decoder settle, or until a predefined stop-criterion is met [3]. After termination, the output $L(\hat{\mathbf{u}})$ of the decoder gives an estimate of the source symbols.

The power of turbo equalization lies in the exchange of extrinsic information $L_e^E(\hat{\mathbf{s}})$ and $L_e^D(\hat{\mathbf{s}}')$ between the equalizer and the decoder. By feeding back the extrinsic information, without creating self-feedback loops, the

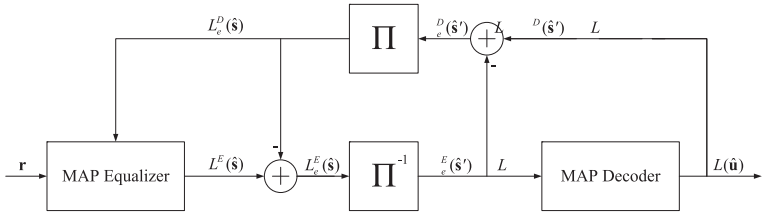

Figure 1 The structure of the turbo equalizer.

correlation between prior information and output information is minimized, allowing the system to converge to an optimal state in the solution space [4,5]. If information is exchanged directly between the equalizer and the decoder by ignoring interleaving and/or extrinsic information, self-feedback loops will be formed. This will cause minimal performance gains, since the equalizer and the decoder will inform each other about information already attained in previous iterations [4].

3 Modeling a turbo equalizer as a quasi-DAG

Suppose a wireless communication system generates a column vector of source bits $\mathbf{s}$ of length N_u and $\mathbf{s}$ is encoded by a convolutional encoder of rate $R_c = 1/n$, producing a coded bit sequence $\mathbf{c}$ of length $N_c = N_u/R_c$. Now suppose that the coded bits sequence is interleaved using a random interleaver, which produces a bit sequence $\acute{\mathbf{c}}$ of length N_c, which is transmitted. The resulting symbol sequence to be transmitted is given by

$$\acute{\mathbf{c}} = \mathbf{J}\mathbf{G}^{\mathbf{T}}\mathbf{s}, \tag{3}$$

where $^{\mathbf{T}}$ denotes the transpose operation and $\mathbf{G}$ is an $N_u \times N_c$ matrix

$$\begin{bmatrix} g_{1K} & \dots & g_{nK} & 0 & \dots & 0 & 0 & 0 & 0 \\ \vdots & \ddots & \vdots & g_{1K} & \dots & 0 & 0 & 0 & 0 \\ g_{11} & \dots & g_{n1} & \vdots & \dots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & g_{11} & \dots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & g_{nK} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & \vdots & g_{1K} & \dots & g_{nK} \\ 0 & 0 & 0 & 0 & \dots & g_{n1} & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & g_{11} & \dots & g_{n1} \end{bmatrix} \tag{4}$$

representing the convolutional encoder, where

$$\mathbf{g} = \begin{bmatrix} g_{11} & \dots & g_{n1} \\ \vdots & \ddots & \vdots \\ g_{1K} & \dots & g_{nK} \end{bmatrix} \tag{5}$$

is the generator matrix of a rate $R_c = k/n$ $(k = 1)$ convolutional encoder with constraint length K and $\mathbf{J}$ is the $N_c \times N_c$ interleaver matrix. Now suppose the symbol sequence $\acute{\mathbf{c}}$ is transmitted over a single-carrier frequency-selective Rayleigh fading channel with a time-invariant IR $\mathbf{h}$ of length L, the received symbol sequence is given by

$$\mathbf{r} = \mathbf{H}\acute{\mathbf{c}} + \mathbf{n}, \tag{6}$$

where $\mathbf{H}$ is the $N_c \times N_c$ channel matrix with the channel IR $\mathbf{h} = \{h_0,\ h_1,\ \dots,\ h_{L-1}\}'$ on the diagonal such that

$$\mathbf{H} = \begin{bmatrix} h_0 & 0 & \dots & 0 & 0 & 0 & 0 \\ \vdots & h_0 & \dots & 0 & 0 & 0 & 0 \\ h_{L-1} & \vdots & \ddots & 0 & 0 & 0 & 0 \\ 0 & h_{L-1} & \ddots & \ddots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & h_0 & 0 & 0 \\ 0 & 0 & 0 & h_{L-1} & \dots & h_0 & 0 \\ 0 & 0 & 0 & 0 & h_{L-1} & \dots & h_0 \end{bmatrix} \tag{7}$$

and $\mathbf{n}$ is a complex Gaussian noise vector with $2N_c$ samples (N_c for real and N_c for imaginary) from the distribution $\mathcal{N}(0, \sigma^2)$.

Figure 2a shows a graphical model of the transmission model in (6), without noise, where it is assumed that $\mathbf{J} = \mathbf{I}$ where $\mathbf{I}$ is an $N_c \times N_c$ identity matrix (i.e., no interleaving is performed) where $R_c = 1/3$ and $L = 2$. It shows that every uncoded bit s_k produces ${R_c}^{-1} = 3$ coded bits $c_{k'}$, $c_{k'+1}$ and $c_{k'+2}$, where $k' = ((k-1)/R_c) + 1$ (k runs from 1 to N_u and k' runs from 1 to N_c). Each received symbol can be expressed as

$$r_{k'} = \sum_{l=0}^{L-1} c_{k'-l} h_l, \tag{8}$$

where $\mathbf{h} = \{h_0,\ h_1\}'$ is the channel IR. Note that h_0 and h_1 are not shown in Figure 2a. This equalization-and-decoding problem can be modeled as a DAG, and the forward–backward algorithm can be used to optimally estimate $\mathbf{c}$, and hence $\mathbf{s}$, with relative ease, since there exists a one-to-one relationship between the observed variables $\mathbf{r}$ and the hidden variables $\mathbf{c}$. There also exists

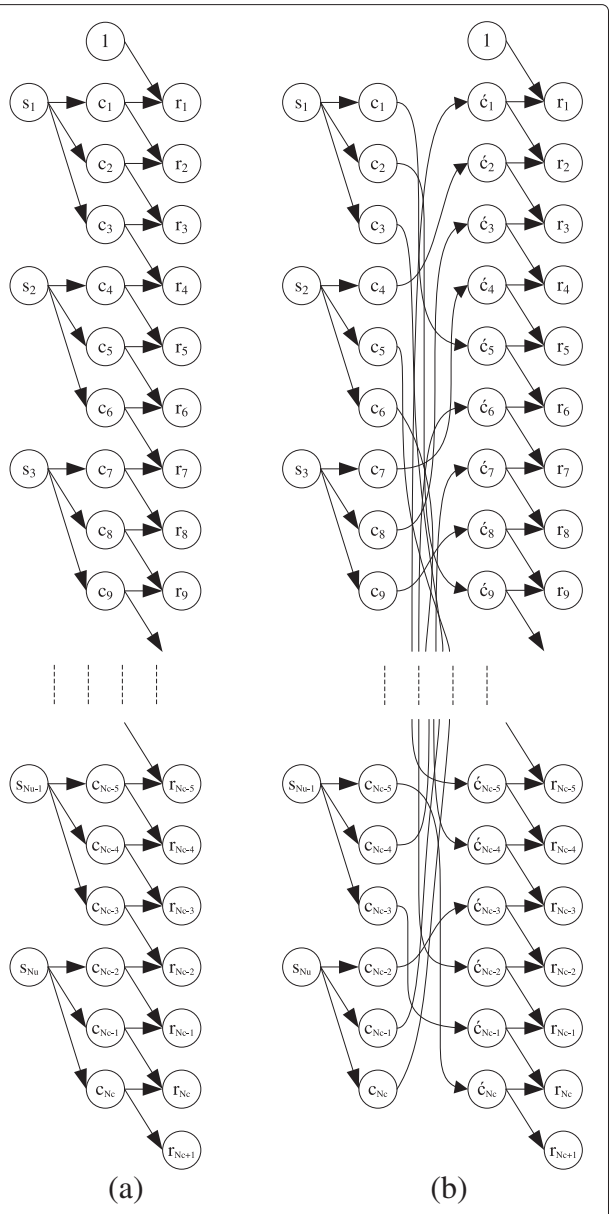

Figure 2 Graphical models of (6) without (a) and with (b) a random interleaver.

a relationship between consecutive codewords (groups of n bits). Figure 2a also depicts the causality relationship between the hidden variables and the observed variables.

Now consider Figure 2b. It shows a graphical model of the transmission model in (6), again without noise, but now $\mathbf{J}$ is a random $N_c \times N_c$ interleaver matrix and again $R_c = 1/3$ and $L = 2$. Each received symbol can be expressed as

$$r_{k'} = \sum_{l=0}^{L-1} \acute{c}_{k'-l} h_l, \tag{9}$$

where $\acute{c}_{k'}$ is the k'th interleaved symbol. It is clear from Figure 2b that there is no obvious relationship between the observed variables $\mathbf{r}$ and the hidden variables $\mathbf{c}$ and that the causality relationship in Figure 2a is destroyed due to the randomization effect of the interleaver. Moreover the relationship between consecutive codewords (groups of n bits) is also destroyed. This problem can therefore no longer be modeled as a DAG and exact inference is in fact impossible [8].

Deinterleaving the received sequence $\mathbf{r}$ will ensure that the one-to-one relationship between each element in $\mathbf{r}$ and $\mathbf{c}$ is restored, but only with respect to the first coefficient h_0 of the IR $\mathbf{h}$. If h_0 is dominant and if $\mathbf{h}$ is sufficiently short, approximate inference is possible due to the negligible effect of h_1 to h_{L-1} on $\mathbf{r}$, but this is not normally the case. In a wireless communication system, transmitting information through a realistic frequency-selective Rayleigh fading channel h_0 cannot be guaranteed to be dominant and the contribution of h_1 to h_{L-1} is not negligible, and therefore this approach will fail. This has been simulated and verified by the authors. Another viable alternative is to model the system as a loopy graph in order to perform approximate inference as in [12], but as stated before, exact inference is impossible [8-10] and full convergence is not guaranteed [13]. In loopy graphs, convergence is normally achieved after many iterations.

The proposed DBN-TE addresses this problem by modeling the turbo equalizer as a quasi-DAG by applying a *transformation* to the ISI-corrupted received symbols, in order to ensure that there will always exist a dominant connection between the hidden variable (codeword symbols) and the observed variable (received symbols) at a given time instant, and only weak connections between the observed variable at the current time instant and hidden variables at past and future time instances. With this transformation the DBN-TE achieves full convergence and is able to perform near-optimal inference in a small number of iterations, in order to estimate the coded sequence $\mathbf{c}$, and hence the uncoded sequence $\mathbf{s}$.

4 Theoretical considerations

In this section, we consider a simplified version of the above-mentioned decoding problem. There are only two hidden and two observation variables, with one weak connection, between the observed value at the time instant 2 and a hidden variable at time instant 1. The purpose of the section is to illustrate, and mathematically analyze, in a clear-cut example how iteration can handle weak connections. Although the analysis in this section is only valid under restrictive assumptions, such as that there is only one weak connection, we end the section with remarks suggesting that it can be generalized.

Consider a directed graph with (hidden) state variables X_1 and X_2, and observables E_1 and E_2 (see Figure 3). We assume that these four random variables are binary.

As already mentioned, we assume that the observable variable E_2 depends not only on the hidden variable X_2, but also on the hidden variable X_1. One way to get back to a classical hidden Markov chain problem is to combine X_1 and X_2 into "super nodes." This increases the size of the state space and hence the computational complexity. We therefore use an iterative method instead, which is explained in Section 5.

We describe the iteration below, but first an informal definition of the weak connection between hidden variable X_1 and observed variable E_2. For the current discussion, it is adequate to understand with a "weak connection" that the conditional probability distribution of the observed variable E_2 given X_2 and $\{X_1 = 0\}$ is close to the distribution given X_2 and $\{X_1 = 1\}$. The answer to the question of how close these two conditional distributions have to be, can in principle be determined from the proof of *Lemma* 1 below. Similar to the preceding section, our diagram is not a tree but a loopy DAG (or quasi-DAG).

To describe the iteration procedure, recall that for a hidden Markov model, the event matrix at position (k, m) is the probability that the observation m is made, given that the value of the hidden state is k. We will modify the forward–backward algorithm (described in Section 5.2.4) to compute the distribution of states given the evidence.

We now describe the iteration. Step 1 is to start with an initial estimate, say (i, j) of the states (X_1, X_2). Step 2 is to modify the forward–backward algorithm by modifying the event matrix that is used at time 2: change the entry in position (k, m) to

$$P(E_2 = m|X_2 = k, X_1 = i). \tag{10}$$

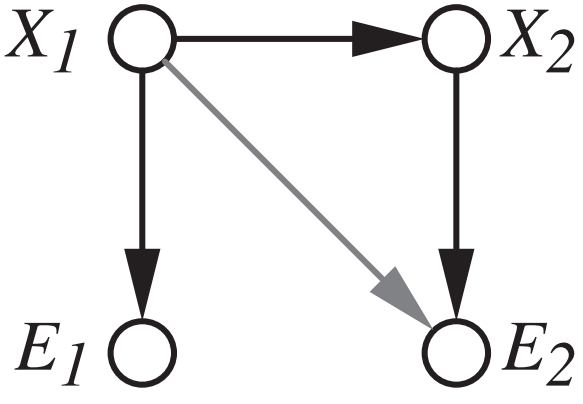

Figure 3 Non-Markovian DAG. We assume that the connection between X_1 and E_2 is "weak".

Use the modified algorithm to compute the distribution of each state given the evidence. Find the most likely states from the distribution.

Step 2 can be iterated several times, but each time with the most recently obtained sequence of states in the place of the initial estimate (i, j). The claim is that these iterations will converge if the connection between E_2 and X_1 is "weak enough."

In the rest of this section, we will interpret what "weak enough" means through notions of real analysis such as metric spaces and $\epsilon - \delta$ description of continuity. The readers unfamiliar with these concepts can skip now to the next section without breaking the flow of their reading.

Note that we can put a metric (distance function) on the collection of event matrices, for example Euclidean distance will do.

The crux of the lemma is illustrated in Figure 4.

Lemma 1. Assume that at all iterations, the modified forward–backward estimates of the probabilities $P(X_1 = 1|E_1 = r, E_2 = s)$ and $P(X_2 = 1|E_1 = r, E_2 = s)$ are different from $\frac{1}{2}$ and 0. Then, if the event matrices, corresponding to the different possibilities for the previous-iterate sequence of bits, are within sufficiently small distance of each other, the iteration procedure will converge to a fixed point; in fact the first and second iterations will already be identical.

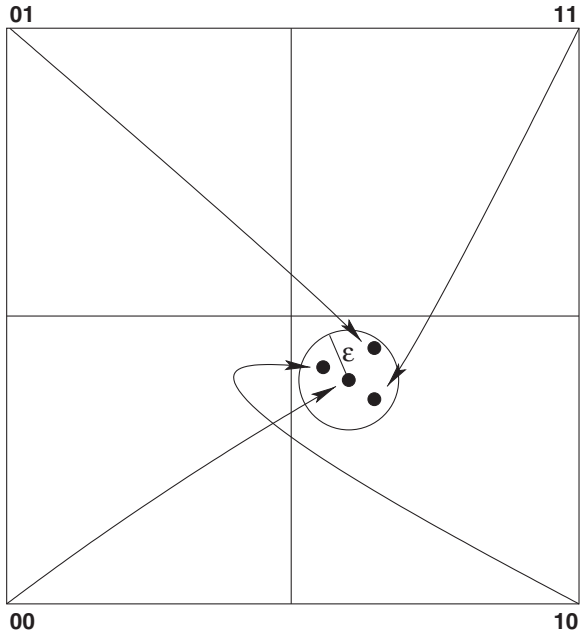

Figure 4 On the four corners we have the possible previous estimates. If the event matrices are "close enough," then all previous estimates will be mapped into the ϵ-ball, and thus into the same quadrant. In a situation where the evidence tables are not "close enough" initially, they may become so after a few iterations.

Proof. It is easy to show that for any fixed observation sequence $(E_1, E_2) = (r, s)$ the dependence, of the posterior probability distribution of (X_1, X_2), on the choice of event matrices, is that of a continuous mapping. We can assume without loss of generality that the iteration is initialized at the state sequence $(0, 0)$, because the argument below will apply to any initialization.

Let (p_1, p_2) be the modified forward–backward estimates obtained, for the probabilities $P(X_1 = 0, X_2 = 0|E_1 = r, E_2 = s)$, when it is initialized at $(0, 0)$. Because these estimates are assumed to be different from $\frac{1}{2}$, the point (p_1, p_2) belongs to the interior of one of the quadrants of the square $[0, 1] \times [0, 1]$. Since it is in the interior, there is an $\epsilon > 0$ so that the ball with midpoint (p_1, p_2) and radius ϵ is contained in the same quadrant.

Our iteration procedure will now move to step 2 and initialize the modified forward–backward algorithm with the output of the first forward–backward run, which is the corner nearest to (p_1, p_2). Now, by the above-mentioned continuity of the posterior distribution on the used event matrices, there is a $\delta > 0$ such that if the all the event matrices are within distance δ of the event matrix corresponding to the above-used initial guess $(0, 0)$, then the output of the modified forward–backward algorithm will be within distance ϵ of (p_1, p_2). Therefore, it will be in the same quadrant as (p_1, p_2). So, rounding to the nearest corner will give the same corner as was yielded by the previous iteration.

We end this section by briefly discussing why this result can be expected to have a version that is also applicable to the less restrictive setup of the previous and next sections.

- *More than two state variables:* With a number of state variables n, the square in the proof will become a hypercube in n dimensions, with 2^n corners, each corner representing a possible previous sequence of estimates of each of the n state variables. The proof will carry over; the $\epsilon - \delta$ description of continuous mappings is still applicable without modification.
- *Event matrices not "close enough:"* If the first estimate of the hidden states is good, it is likely that it will be mapped to a quadrant, of which the corner will be mapped to the right quadrant in the next iteration. In this case more than one iteration will be needed.
- *More weak connections:* With more than two state variables, it becomes possible that E_k is connected with X_l for some $l < k - 1$. The entries of the event tables will then have to be set to a probability that is conditioned on a sequence longer than that appearing in equation (10). This does not affect the proof, as long as the connections are weak enough. Here the connections being weak enough means that the event matrices that are possible due to different possible observation histories, are all close enough in the sense of the lemma.

□

5 DBN turbo equalizer

In this section, the DBN-TE is discussed. The first part explains the transformation that is applied to the channel matrix in order to ensure that there exists a dominant connection between the observed variables and their corresponding hidden variables, and the second part explains the operation of the DBN-TE, which jointly models the equalization and decoding stages on a quasi-DAG as defined in the previous section.

5.1 Transformation

For this exposition, assume that the coded symbols $\mathbf{c}$ are transmitted through a channel $\mathbf{Q} = \mathbf{HJ}$, where $\mathbf{H}$ is the channel matrix and $\mathbf{J}$ is the interleaver matrix as previously defined. Therefore, Equation (6) can be written as

$$\mathbf{r} = \mathbf{Qc} + \mathbf{n}. \tag{11}$$

For the DBN-TE to perform approximate inference there must exist a strong connection between the observed variable and the hidden variable at time instance k, and weak connections must exist between the observed variable at time instance k and hidden variables at other time instances. The randomization effect of the interleaver must also be mitigated in order for the turbo equalizer to be modeled as a quasi-DAG so that there can exist a one-to-one relationship (dominant connection) between the observed variable and the corresponding hidden variable at time instance k.

To ensure that the connections between the observed variable and neighboring hidden variables are weak, the energy must be concentrated in the first tap h_o of $\mathbf{h}$. This can be achieved by applying a minimum phase prefilter to the received symbols $\mathbf{r}$ [14]. This process produces a filtered received symbol sequence and a minimum phase channel IR.

In order to model the turbo equalization problem as a quasi-DAG, the randomization effect of the random interleaver must be mitigated. Figures 5 and 6 show $\|\mathbf{Q}\|$ for systems with a channel IR lengths of $L = 1$ and $L = 3$, respectively, for a hypothetical system with parameters $N_u = 50$, $N_c = 150$, $R_c = 1/3$ at a mobile speed of 3 km/h and no frequency hops. It should be clear that any sequence $\mathbf{c}$ that is transmitted through a channel $\mathbf{Q}$, as described in (11), will be subject to randomization.

To mitigate the effect of the interleaver, the following transformation is applied to $\mathbf{r}$:

$$\mathbf{Q}^H\mathbf{r} = \mathbf{Q}^H\mathbf{Qc} + \mathbf{Q}^H\mathbf{n}, \tag{12}$$

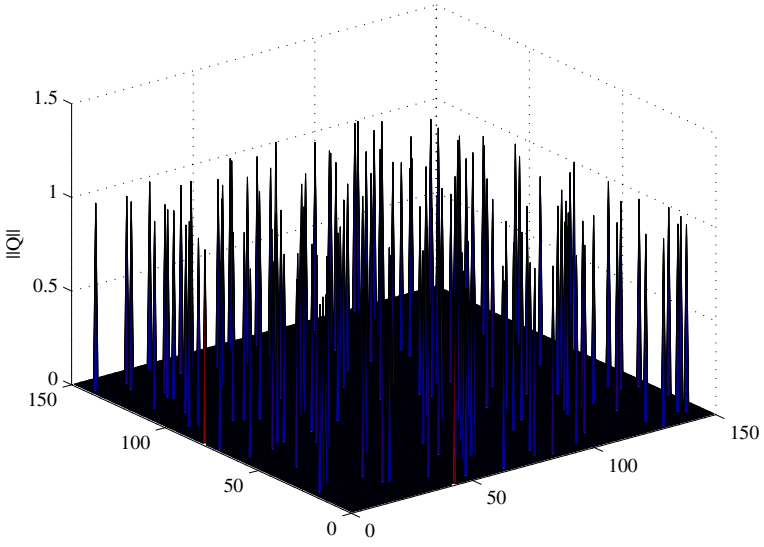

Figure 5 $\|\mathbf{Q}\|$ for a system with $L = 1$ IR coefficients.

which is equivalent to transmitting the coded symbol sequence $\mathbf{c}$ through a channel $\mathbf{U}$, where

$$\mathbf{U} = \mathbf{Q}^H\mathbf{Q}, \tag{13}$$

so that

$$\mathbf{Q}^H\mathbf{r} = \mathbf{U}\mathbf{c} + \mathbf{Q}^H\mathbf{n}. \tag{14}$$

Figures 7 and 8 show $\|\mathbf{U}\|$ for systems with channel IR lengths of $L = 1$ and $L = 3$, respectively. It is clear that this transformation mitigates the randomness exhibited in $\mathbf{Q}$, since the new "channel" $\mathbf{U}$ is diagonally dominant. The one-to-one relationship between the observed variables and the corresponding hidden variables are therefore restored. Minimum-phase filtering of $\mathbf{r}$ is performed before performing the transformation in (12).

Therefore, applying a minimum-phase filter to $\mathbf{r}$ and performing the transformation in (14), all the conditions are met to model the turbo equalizer as a quasi-DAG with dominant connections between the observed variables and their corresponding hidden variables. Minimum-phase filtering ensures that a dominant connection exists between the observed variable and the hidden variable at time instance t, and that there exist weak connections

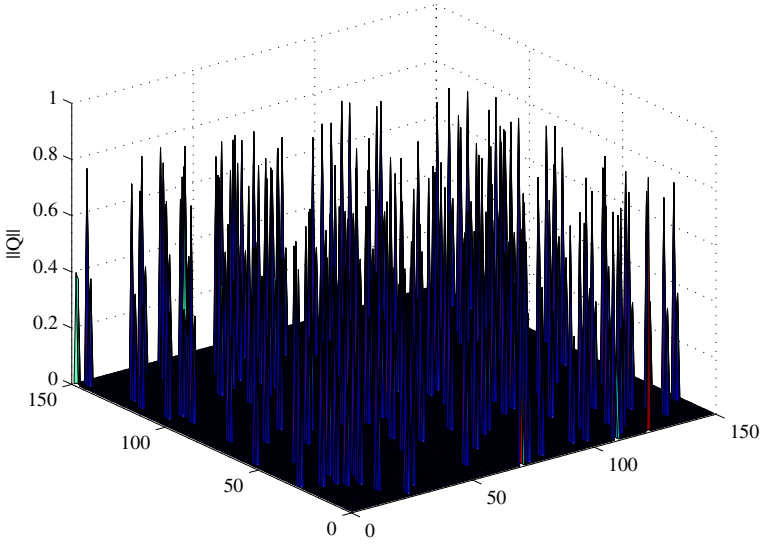

Figure 6 $\|\mathbf{Q}\|$ for a system with $L = 3$ IR coefficients.

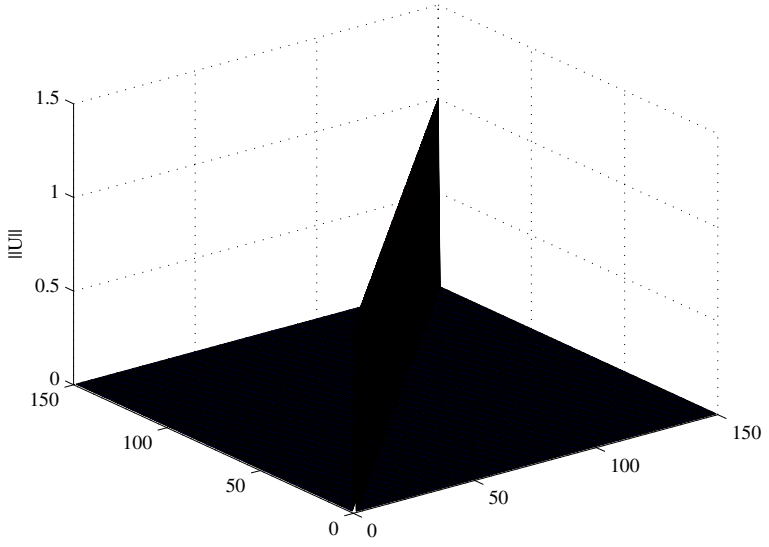

Figure 7 $\|\mathbf{U}\|$ for a system with $L = 1$ IR coefficients.

between the observed variable at time instance t and the hidden variable at other time instances, while the transformation in (14) mitigates the randomization effect of the interleaver so that there exists a one-to-one relationship between each observed variable and its corresponding hidden variable. By performing the transformation described here, all conditions for convergence as described in Section 4 are met.

5.2 The DBN-TE algorithm

After making preparation for the turbo equalization problem to be modeled as a quasi-DAG, as explained in the previous section, the DBN-TE algorithm can be executed. In this section, various aspects of the DBN-TE are discussed after which a step-by-step summary is provided in Section 5.2.8 in the form of a pseudocode algorithm, encapsulating the working of the DNB-TE.

We assume a system with a uncoded block length of N_u, using the rate $R_c = 1/3$, constraint length $K = 3$, convolutional encoder in Figure 9 to produce N_c bits, where $N_c = N_u/R_c$. The coded bits are interleaved with a random interleaver and passes through a multipath channel with an IR of length L.

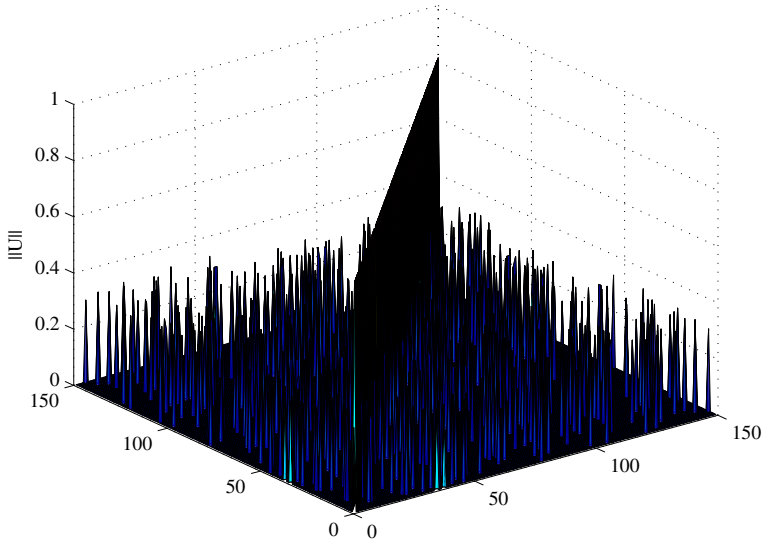

Figure 8 $\|\mathbf{U}\|$ for a system with $L = 3$ IR coefficients.

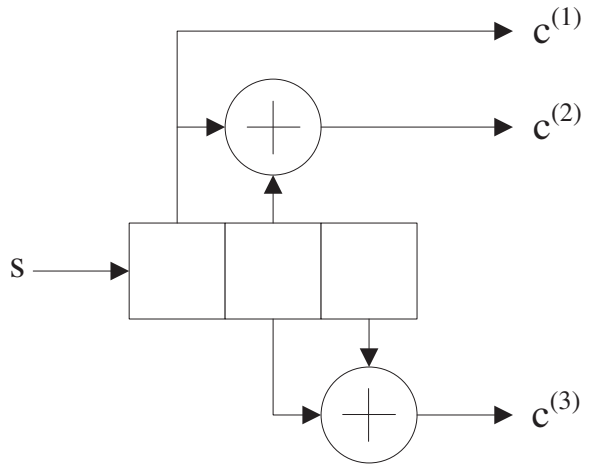

Figure 9 Rate $R_c = 1/3$ convolutional encoder.

5.2.1 Graph construction

The graph is constructed to model the possible outputs of a convolutional encoder in much the same way as a trellis is constructed in a conventional MAP decoder. For the DBN-TE graph the number of states per time instance t is equal to the number of possible *state transitions*, given by $M = 2^K$, where K is the encoder constraint length. This is different from the number of states in a MAP decoder, which is equal to the number of possible *states*, given by $M = 2^{K-1}$. The number of time instances in the DBN-TE graph is equal to the number of uncoded bits N_u, which is also equal to the number of codewords. Figure 10a,b shows the graphical model of a DBN-TE for the first and subsequent iterations, respectively, where the dashed lines in Figure 10b depict weak connections due to ISI. Each $\mathbf{X_t}$ on the graph contains a set of M state transitions $x_t^{(m)}$, where $t = 1, 2, \ldots, N_u$ and $m = 1, 2, \ldots, M$.

During the first iteration no coded bit estimates $\tilde{\mathbf{c}}$ are available, so the graphical model is a pure DAG due to the fact that the current state is only dependent on the previous state. Hence only $U_{t,t}$ is used in the cost function of the sensor model, where $U_{t,t}$ is a coefficient on the diagonal of the new channel matrix $\mathbf{U}$ in (14). After the first iteration estimates of the coded bits $\tilde{\mathbf{c}}$ are produced and can therefore be used in subsequent iterations. During subsequent iterations then, $U_{t,u}$ and $U_{t,v}$ are also considered in the cost function of the sensor model, where $u = 1, 2, \ldots, t-1$ and $v = t+1, t+2, \ldots, N_u$.

5.2.2 State transition output table

The output associated with each state transition is also tabulated using the encoder state diagram in Figure 11. The output produced by each state transition $x_t^{(m)}$, $m = 1, 2, \ldots, 8$, is determined by loading the bit-values of the current state into the leftmost $K - 1 = 2$ fields of the convolutional encoder in Figure 9, and then placing a 0 and a

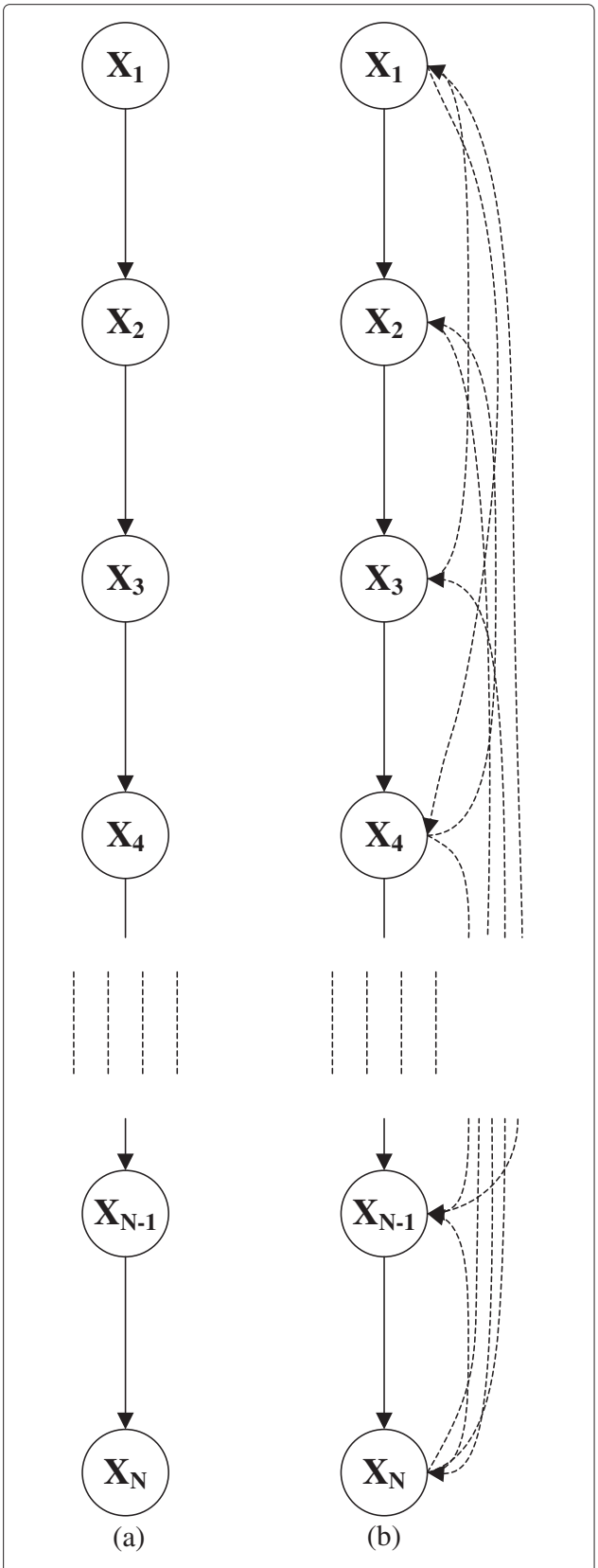

Figure 10 Graphical models for the (a) first iteration and (b) subsequent iterations.

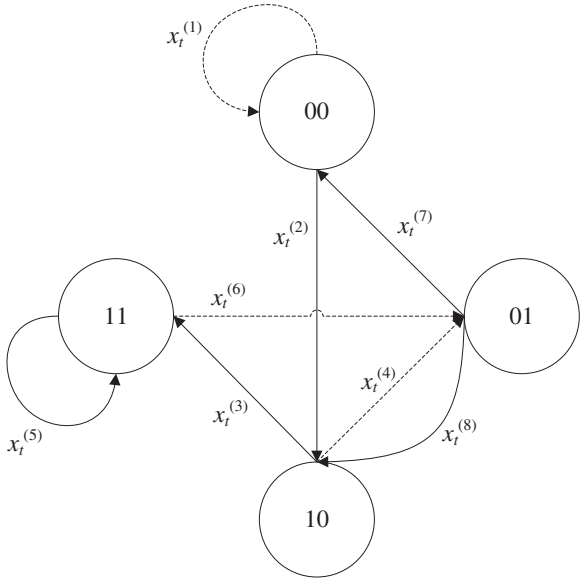

Figure 11 Convolutional encoder state diagram.

1, respectively, on the input of the encoder, each producing a new codeword $c^{(1)}c^{(2)}c^{(3)}$ at the output. This process is followed exhaustively and tabulated. Table 1 shows the state transition outputs of the encoder in Figure 9 that results from moving from one state to the next.

5.2.3 Transition probability table

The DBN-TE depends on a transition model to describe the permissible state transitions. This is constructed by examining the encoder state transition diagram in Figure 11 and noting the possible state transitions. The solid lines and dashed lines indicate state transitions caused by ones and zeros at the input of the encoder. It is clear from Figure 11 that only two state transitions emanate from any given state transition (one caused by a 1 at the input and one caused by a 0 at the input). Table 2 shows the transition probabilities of the encoder in Figure 9 of which the state transition diagram is shown in Figure 11.

Table 1 State transition output table

	$c^{(1)}$	$c^{(2)}$	$c^{(3)}$
$x_t^{(1)}$	0	0	0
$x_t^{(2)}$	1	1	0
$x_t^{(3)}$	1	0	1
$x_t^{(4)}$	0	1	1
$x_t^{(5)}$	1	0	0
$x_t^{(6)}$	0	1	0
$x_t^{(7)}$	0	0	1
$x_t^{(8)}$	1	1	1

5.2.4 The forward–backward algorithm

The forward–backward algorithm computes the distribution over past states given evidence up to the present [15]. It determines the exact MAP distribution $\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:t})$ for $1 \leq k < t$, where $\mathbf{e}_{1:t}$ is a sequence of observed variables from time 1 to t. This is done by calculating two evidence "messages"—the forward message from 1 up to k and the backward message from $k+1$ up to t.

Forward message The forward message computes the posterior distribution over the future state, given all evidence up the current state. To compute the forward message, the current state is projected forward from time t to time $t+1$ and is then updated using the new evidence $\mathbf{e}_{t+1}$. To obtain the prediction of the next state it is necessary to condition on the current state $\mathbf{X}_t$, hence:

$$\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t+1}) = \alpha\mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1}) \sum_{\mathbf{x}_t} \mathbf{P}(\mathbf{X}_{t+1}|\mathbf{x}_t)P(\mathbf{x}_t|\mathbf{e}_{1:t}) \tag{15}$$

where α is a normalization constant, $\mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1})$ is obtained from the sensor model, $\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{x}_t)$ is the transition model and $P(\mathbf{x}_t|\mathbf{e}_{1:t})$ is the current state distribution. The forward message can be computed recursively using (15).

Backward message The backwards message is computed in a similar fashion. It computes the posterior distribution over past state, given all future evidence up to the current state. Whereas the forward message is computed forwards from 1 to k, the backwards message is computed backwards from t to $k+1$. Thus, the backwards message determines

$$\mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k) = \sum_{\mathbf{x}_{k+1}} \mathbf{P}(\mathbf{e}_{k+1}|\mathbf{x}_{k+1})\mathbf{P}(\mathbf{x}_{k+1}|\mathbf{X}_k)P(\mathbf{e}_{k+2:t}|\mathbf{x}_{k+1}) \tag{16}$$

Table 2 Transition probability table; transition probability table showing $P(x_t^{(m)}|x_{t+1}^{(n)})$

	$x_{t+1}^{(1)}$	$x_{t+1}^{(2)}$	$x_{t+1}^{(3)}$	$x_{t+1}^{(4)}$	$x_{t+1}^{(5)}$	$x_{t+1}^{(6)}$	$x_{t+1}^{(7)}$	$x_{t+1}^{(8)}$
$x_t^{(1)}$	$\frac{1}{2}$	$\frac{1}{2}$	0	0	0	0	0	0
$x_t^{(2)}$	0	0	$\frac{1}{2}$	$\frac{1}{2}$	0	0	0	0
$x_t^{(3)}$	0	0	0	0	$\frac{1}{2}$	$\frac{1}{2}$	0	0
$x_t^{(4)}$	0	0	0	0	0	0	$\frac{1}{2}$	$\frac{1}{2}$
$x_t^{(5)}$	0	0	0	0	$\frac{1}{2}$	$\frac{1}{2}$	0	0
$x_t^{(6)}$	0	0	0	0	0	0	$\frac{1}{2}$	$\frac{1}{2}$
$x_t^{(7)}$	$\frac{1}{2}$	$\frac{1}{2}$	0	0	0	0	0	0
$x_t^{(8)}$	0	0	$\frac{1}{2}$	$\frac{1}{2}$	0	0	0	0

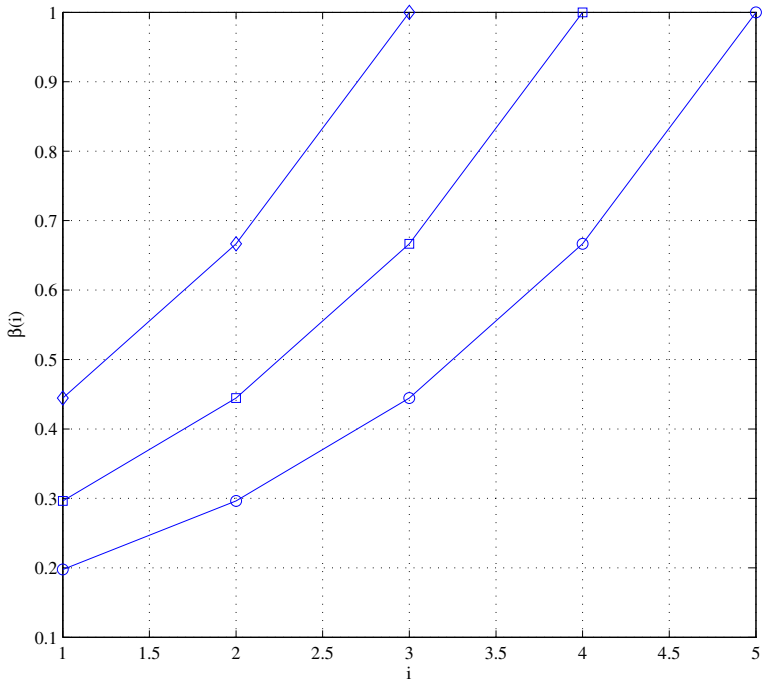

Figure 12 The output of function $\beta(i)$ for Z=3, Z=4 and Z=5 iterations. Blue diamond, $Z = 3$; Blue square, $Z = 4$; Blue circle, $Z = 5$.

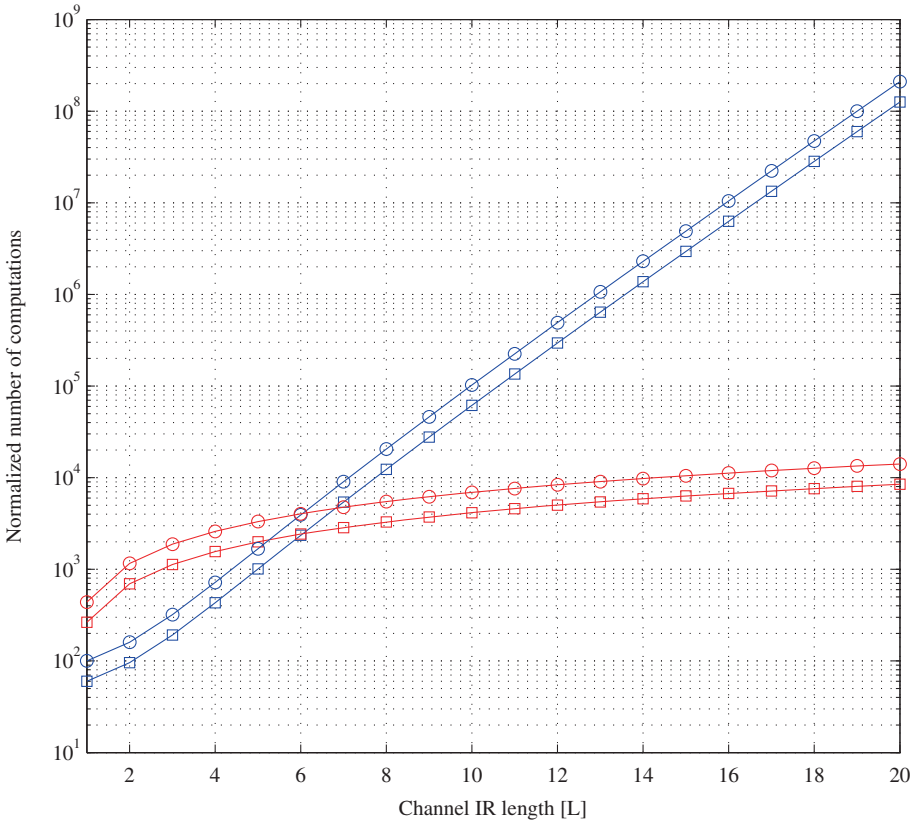

Figure 13 DBN-TE and CTE normalized computational complexity for different channel IR lengths and different numbers of iterations. Blue square, CTE - $Z = 3$; Blue circle, CTE - $Z = 5$; Red square, DBN-TE - $Z = 3$; Red circle, DBN-TE - $Z = 5$.

where $\mathbf{P}(\mathbf{e}_{k+1}|\mathbf{x}_{k+1})$ is obtained from the sensor model, $\mathbf{P}(\mathbf{x}_{k+1}|\mathbf{X}_k)$ is the transition model and $P(\mathbf{e}_{k+2:t}|\mathbf{x}_{k+1})$ is the current state distribution. The backward message can be computed recursively using (16).

Forward–backward message Finally, by combining the forward and backward message, the posterior distribution over all states at any time instance $1 \leq k < t$ can be determined as

$$\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:t}) = \alpha\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:k})\mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k). \tag{17}$$

5.2.5 Sensor model

The conditional probabilities obtained from the sensor model for the respective forward and backward messages, $\mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1})$ and $\mathbf{P}(\mathbf{e}_{k+1}|\mathbf{x}_{k+1})$ are determined by calculating a metric between the observed variable $\mathbf{e}_t$ and the hidden variable $x_t^{(m)}$, where $m = 1, 2, \ldots, M$. The observed variable $\mathbf{e}_t$ consists of ${R_c}^{-1}$ received symbols $r_{t'+1}$ to $r_{t'+{R_c}^{-1}}$, where $t' = ((t-1)/R_c) + 1$ (t runs from 1 to N_u and t' runs from 1 to N_c), and the hidden variable $x_t^{(m)}$ consists of the output ($c^{(1)}$, $c^{(2)}$, and $c^{(3)}$) associated with state transition $x_t^{(m)}$ in Table 1.

During the first iteration, only the dominant connection $U_{t',t'}$ between the observed variable $\mathbf{e}_t$ and the hidden variable $\mathbf{x}_t$ is used in the cost calculation, since no coded bit estimates $\tilde{\mathbf{c}}$ are available at that point. Recall that $\mathbf{U}$ is the new channel matrix brought about by performing the transformation in (12), resulting in the new transmission model in (14) having the desired properties to model the system as a quasi-DAG. During the first iteration the system can therefore be modeled as a pure DAG, as if no ISI occurred. Given a hidden variable or state transition output x_{t+1}^n, its associated bits (as tabulated in Table 1) are used together with observed variables/received symbols $r_{t'+1}$ to $r_{t'+{R_c}^{-1}}$, where $t' = ((t-1)/R_c) + 1$ and ${R_c}^{-1}$ is the number of encoder output bits, to calculate the cost of the nth state transition at time instance t

$$\Delta_t^n = \sum_{j=0}^{R^{-1}-1} \|r_{t'+j} - U_{t'+j,t'+j} c_n^{j+1} \beta(i)\|^2, \tag{18}$$

where $t = 1, 2, \ldots, N_u$ runs over time for the uncoded bit estimates and $\beta(.)$ is a function that produces an optimization scaling factor for each iteration i. $\mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1})$ in (15)

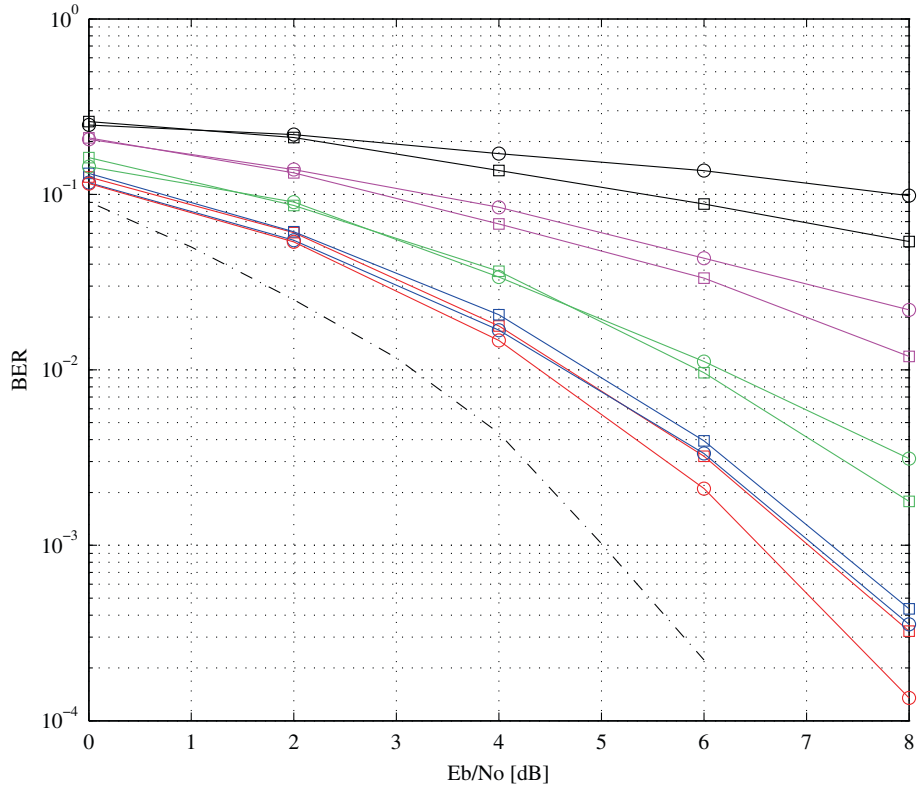

Figure 14 DBN-TE and CTE performance at different mobile speeds in a system with channel IR length $L = 6$. Black circle, CTE: 110 km/h; Black square, DBN-TE: 110 km/h; Pink circle, CTE: 80 km/h; Pink square, DBN-TE: 80 km/h; Green circle, CTE: 50 km/h; Green square, DBN-TE: 50 km/h; Blue circle, CTE: 20 km/h; Blue square, DBN-TE: 20 km/h; Red circle, CTE: 3 km/h; Red square, DBN-TE: 3 km/h; Black dashed, Decoded AWGN.

is determined by

$$\mathbf{P}(\mathbf{e}_{t+1}|x_{t+1}^n) = \alpha \exp(-\Delta_t^n/2\sigma^2) \tag{19}$$

where α is a normalization constant and σ is the noise standard deviation.

During subsequent iterations, LLR estimates of the uncoded bits are available, since the first set of LLRs are produced after the first iteration. Therefore, the system can be modeled as a quasi-DAG due to the fact that there exists a dominant connection between the observed variable $\mathbf{e}_t$ and the hidden variable $\mathbf{x}_t$ and weak connections between the observed variable $\mathbf{e}_t$ and other hidden variables. Thus we also use the rest of the coefficients $U_{t',1}$ to U_{t',N_c} (and not only $U_{t',t'}$) in the cost calculation. Analogous to the first iteration, the output bits associated with a given state transition x_{t+1}^n are used together with observed variables $r_{t'+1}$ to $r_{t'+R_c{}^{-1}}$ *as well as* the LLR estimates $\tilde{\mathbf{c}}$ of the uncoded bits to calculate the cost of the nth state transition at time instance t. Therefore, the cost of the nth state transition for subsequent iterations at time instance t is given by

$$\begin{aligned}\Delta_t^n = &\sum_{j=0}^{R_c{}^{-1}+1} \|r_{t'+j} - U_{t'+j,t'+j} c_n^{j+1} \\ &- \sum_{v=1,v\neq t',\|U_{t'+j,v}\|>0}^{N_c} U_{t'+j,v}\tilde{c}_v\beta(i)\|^2.\end{aligned} \tag{20}$$

The last term in (20) contains the ISI terms that must be subtracted from the received symbols in order to minimize Δ_t^n so that $\mathbf{P}(\mathbf{e}_{t+1}|x_{t+1}^n)$, determined as in (19) can be maximized.

5.2.6 Computing LLR estimates

After the forward and backward messages are combined as in (17), the LLRs for each uncoded bit is determined from the graph. $R_c{}^{-1}$ LLR vectors of length N_u are determined—each one corresponding to one output bit of the encoder—after which they are multiplexed to form one vector of length N_c containing the LLR estimates $\tilde{\mathbf{c}}$ of the coded bits $\mathbf{c}$. With reference to the state transitions in Table 1, the LLRs for the convolutional encoder in Figure 9, are determined as follows:

$$\tilde{\mathbf{c}}^{(1)} = \log\left(\frac{\sum_{j=1,c^{(1)}=1}^{M}\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:t})}{\sum_{j=1,c^{(1)}=0}^{M}\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:t})}\right) \tag{21}$$

$$\tilde{\mathbf{c}}^{(2)} = \log\left(\frac{\sum_{j=1,c^{(2)}=1}^{M}\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:t})}{\sum_{j=1,c^{(2)}=0}^{M}\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:t})}\right) \tag{22}$$

$$\tilde{\mathbf{c}}^{(3)} = \log\left(\frac{\sum_{j=1,c^{(3)}=1}^{M}\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:t})}{\sum_{j=1,c^{(3)}=0}^{M}\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:t})}\right) \tag{23}$$

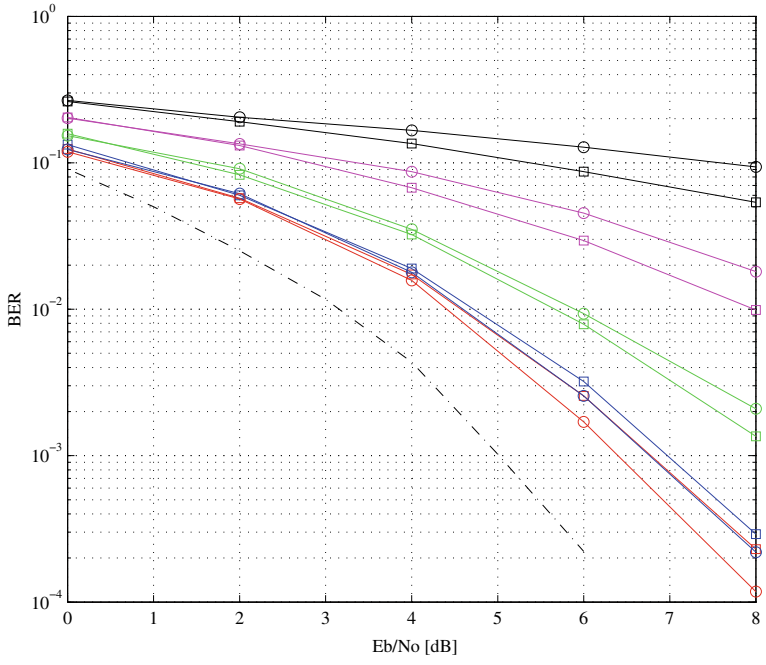

Figure 15 DBN-TE and CTE performance at different mobile speeds in a system with channel IR length $L = 8$. Black circle, CTE: 110 km/h; Black square, DBN-TE: 110 km/h; Pink circle, CTE: 80 km/h; Pink square, DBN-TE: 80 km/h; Green circle, CTE: 50 km/h; Green square, DBN-TE: 50 km/h; Blue circle, CTE: 20 km/h; Blue square, DBN-TE: 20 km/h; Red circle, CTE: 3 km/h; Red square, DBN-TE: 3 km/h; Black dashed: Decoded AWGN.

The final LLR vector is constructed by multiplexing the respective LLR vectors such that

$$\tilde{\mathbf{c}} = \{\tilde{c}_1^{(1)}, \tilde{c}_1^{(2)}, c_1^{(3)}, \tilde{c}_2^{(1)}, \tilde{c}_2^{(2)}, \tilde{c}_2^{(3)}, \ldots, \tilde{c}_{N_u}^{(1)}, \tilde{c}_{N_u}^{(2)}, \tilde{c}_{N_u}^{(3)}\} \quad (24)$$

which is used in (20) in the next DBN-TE iteration.

5.2.7 Optimization

To improve the BER performance of the system, simulated annealing is used [15]. Simulated annealing is usually used in neural networks to allow the network to escape suboptimal basins of attraction in order to converge to a near-optimal solution in the solution space. Since the DBN-TE employs a soft-feedback mechanism, simulated annealing can also be applied to the coded symbol estimates $\tilde{\mathbf{c}}$ that are fed back after the first iteration, thus allowing the DBN-TE to converge to a state where the BER performance is near-optimal. The optimization scaling function $\beta(.)$ in (18) and (20)

$$\beta(i) = 1.5^{(i-Z)/Z}, \quad (25)$$

where Z is the number of iterations, is updated with each iteration i, always starting at $0 < \beta(1) << 1$ for the first iteration ($i = 1$) and finishing at $\beta(Z) = 1$ for the final iteration ($i = Z$). Figure 12 shows $\beta(i)$ for $Z = 3$, $Z = 4$ and $Z = 5$. Simulation results in Section 7 show the performance of the DBN-TE with and without simulated annealing.

5.2.8 Pseudocode algorithm

Additional file 1a–e shows the pseudocode of the DBN-TE algorithm. The DBN-TE algorithm iteratively computes the forward–backward message before producing LLR estimates of the coded symbols. After the final iteration the estimated turbo equalized uncoded information bits are returned.

Function definitions DBN_TE receives as input the received ISI-corrupted coded symbols, the transition probability table, the transition output table, the codeword length, the coded data block length, the number of states of the graph, the new channel after transformation, the number of iterations and the noise standard deviation.

FORWARD_MESSAGE and BACKWARD_MESSAGE receive as input the same variable as DBN_TE, accept for the number of iterations Z, and additionally they also received the LLR estimates and the optimization scaling factor as input.

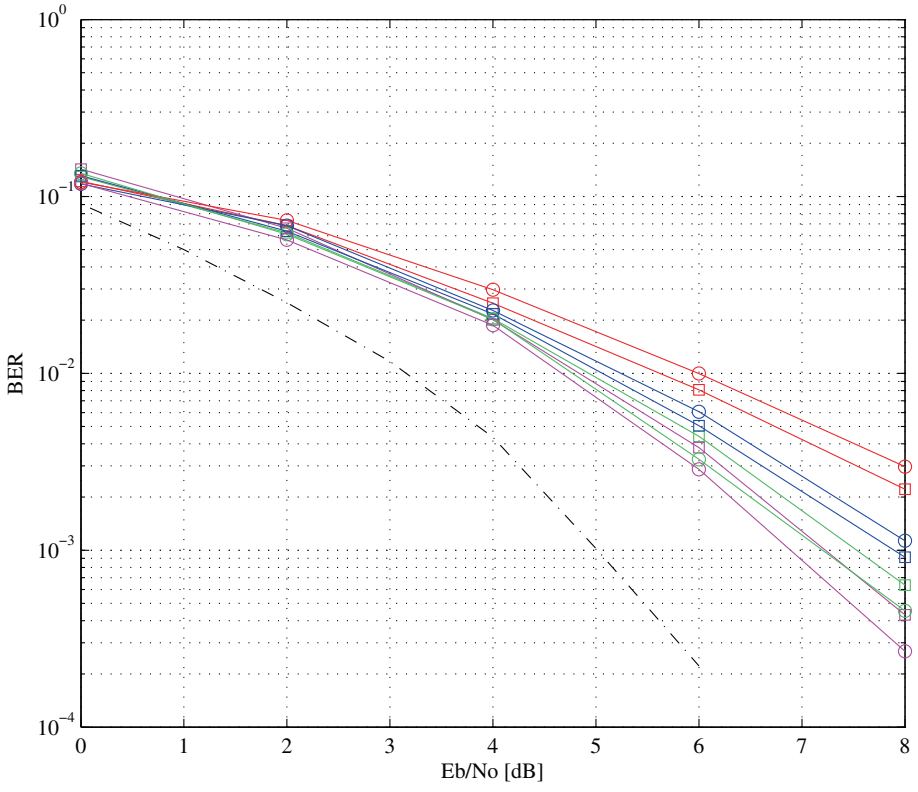

Figure 16 DBN-TE and CTE performance for different numbers of frequency hops in a system with channel IR length $L = 6$. Red circle, CTE: No hops; Red square, DBN-TE: No hop; Blue circle, CTE: 2 hops; Blue square, DBN-TE: 2 hops; Green circle, CTE: 4 hops; Green square, DBN-TE: 4 hops; Pink circle, CTE: 8 hops; Pink square, DBN-TE: 8 hops; Black dashed, Decoded AWGN.

FORWARD_BACKWARD_MESSAGE receives only the result of FORWARD_MESSAGE and BACKWARD_MESSAGE, while LLR_ESTIMATES takes as input the forward–backward message resulting from FORWARD_BACKWARD_MESSAGE as well as the state transition output table.

Optimization scaling factor For each iteration a new optimization scaling factor is produced, as explained earlier. BETA implements (25) and returns the scaling factor used in the calculation of the forward and backward messages.

Forward message The algorithm starts by initializing the forward message, based on the initial state of the encoder shift register. Since it is assumed that the encoder shift register always starts in the all-zero state, the forward message is initialized accordingly, using the appropriate entries in the transition probability table (Table 2). The forward message is therefore initialized as

$$forward(1,:) = trans\_prob\_table(1,:), \tag{26}$$

such that $forward(1,:) = [0.5, 0.5, 0, 0, 0, 0, 0, 0]'$ since only state transitions $x_{t+1}^{(1)}$ and $x_{t+1}^{(2)}$ can emanate from state transitions $x_t^{(1)}$ and $x_t^{(7)}$, which lead to the all-zero state (see Figure 11).

The forward message is calculated next by iterating over time from $k = 2$ to $k = N$ while iterating over M states $m = 1$ to $m = M$ for each k. For each (k, m) pair, the forward message is initialized to zero, after which the message is updated by multiplying and accumulating messages from the previous time-step $k - 1$. The forward messages from the previous time-step $k - 1$ are multiplied with their respective transition probabilities (determined by the current state m) and summed together to form the new forward message at time-step k (at state m in the graph). Up to this point the forward message contains the collective contribution of the state distributions of the previous states, corresponding to the terms inside the summation in (15).

To include the evidence at graph state (k, m), the cost of the state transition associated with state m in the graph at time k must be calculated. The GET_COST functions takes as input the received codeword, LLR estimates of the codeword, the new channel **U**, the optimization scaling factor and the noise standard deviation, and determines the cost as in (18) for the first iterations and (20) for subsequent iterations. The forward message is updated

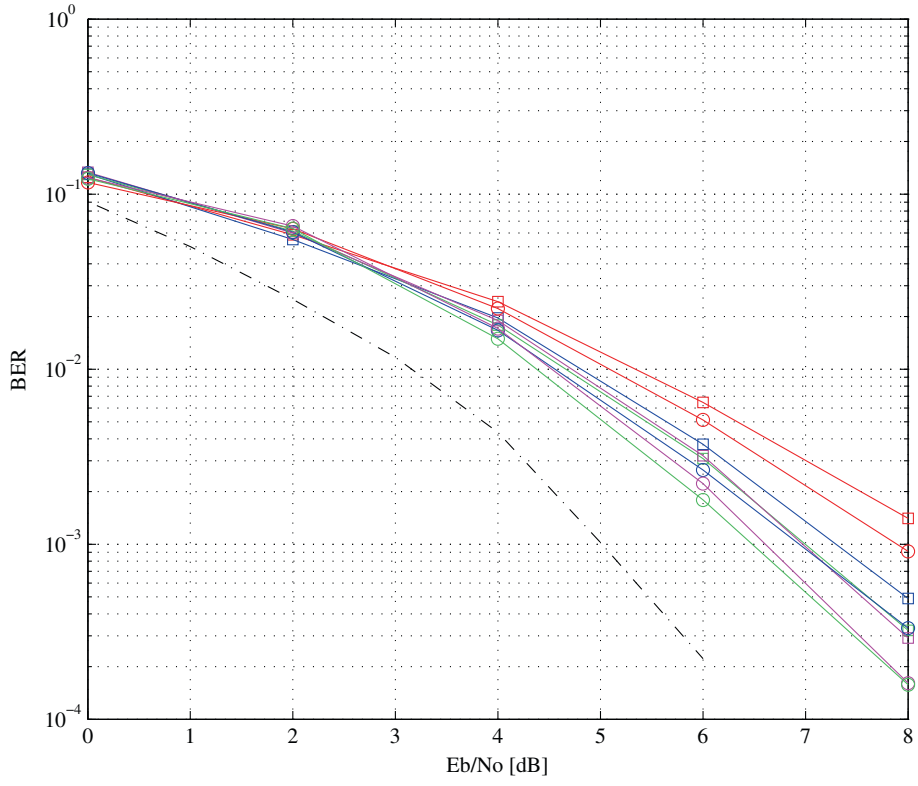

Figure 17 DBN-TE and CTE performance for different numbers of frequency hops in a system with channel IR length $L = 8$. Red circle, CTE: No hops; Red square, DBN-TE: No hop; Blue circle, CTE: 2 hops; Blue square, DBN-TE: 2 hops; Green circle, CTE: 4 hops; Green square, DBN-TE: 4 hops; Pink circle, CTE: 8 hops; Pink square, DBN-TE: 8 hops; Black dashed, Decoded AWGN.

by multiplying with the output of the normal probability distribution function NORMAL_PDF (implemented as in (19)) which takes the cost and noise standard deviation as input. This completes the forward message. It now fully corresponds to (15). After computing M forward messages at time-step k (one for each of the M states at time-step k), all the messages at time-step k are normalized (NORMALIZE) so as to prevent message values from becoming very small due to multiplication of probabilities, when large data block sizes are used.

Backward message The backward message is initialized similar to the forward message, based on the final state of the encoder. The encoder is forced into the all-zero state and the end of the transmitted data block, and hence the backward message is initialized as

$$backward(N,:) = trans\_prob\_table(:,1)', \tag{27}$$

such that $backward(N,:) = [0.5, 0, 0, 0, 0, 0, 0.5, 0]'$ since the state transitions emanating from the all-zero state, $x_{t+1}^{(1)}$ and $x_{t+1}^{(2)}$, are preceded by state transitions $x_t^{(1)}$ and $x_t^{(7)}$.

The backward message is calculated in a similar fashion as the forward message, accept that iteration over time starts at $k = N - 1$ and ends at $k = 1$. Also note that the cost is not calculated for the current graph state (k, m), but for all those preceding it (at time $k + 1$). Note the sensor model output is *inside* the summation in (16), whereas it is outside the summation in (15). The backward message is therefore calculated by accumulating information at the current graph state (at time k) from preceding graph states (at time $k + 1$). The sensor model in (19) is applied (NORMAL_PDF) to each preceding state, multiplied with the transition probability connecting the current state with the preceding state, and then multiplied with the message that corresponds to the preceding state. This result is then summed together for all M preceding states and stored at the current state.

Forward–backward message The forward backward message is created by multiplying each corresponding forward and backward message value for each (k, m) pair, and normalizing the results as before.

Calculate LLR estimates The LLR estimates are calculated in three phase to produce a sequence of N LLR estimates for each output bit of the decoder (assuming the rate $R_c = 1/3$ convolutional encoder in Figure 9), after

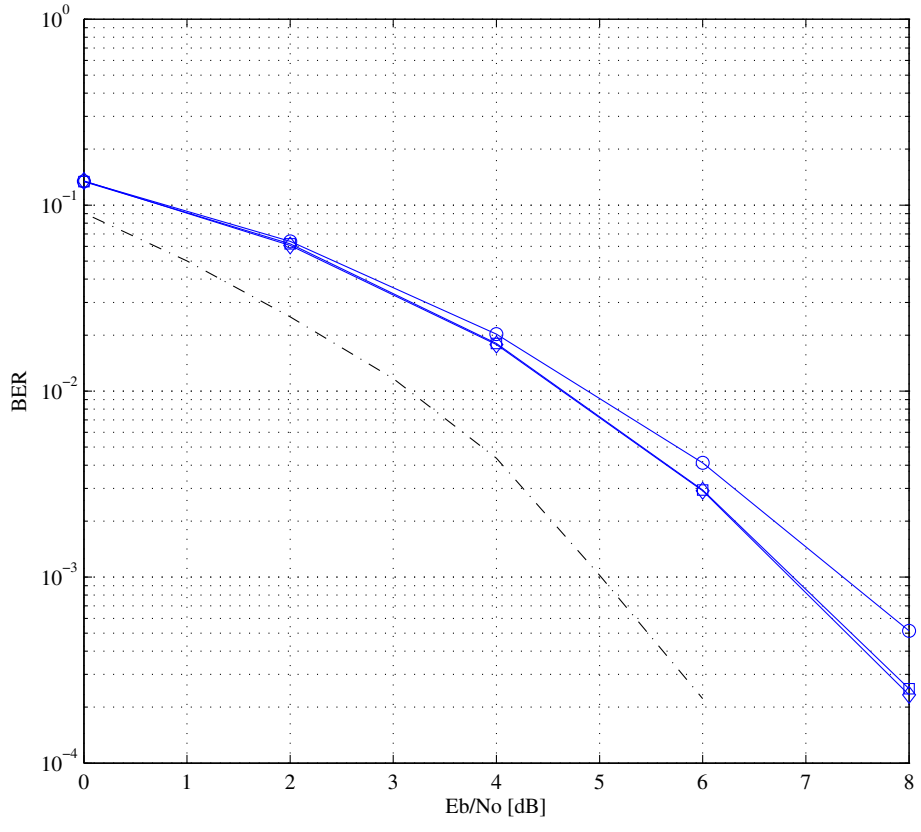

Figure 18 DBN-TE performance in a system with long channel IR lengths. Blue circle, DBN-TE - $L = 5$; Blue square, DBN-TE - $L = 10$; Blue diamond, DBN-TE - $L = 20$; Black dashed, Decoded AWGN.

which the three LLR sequences are multiplexed to create a sequence of LLR estimates of length N/R_c. The LLR vectors are calculated by noting the ones and zeros in Table 1 that correspond to the respective output bits generated by the encoder. For instance, the first output bit $c^{(1)}$ in Table 1 is one for state transitions $x_t^{(2)}$, $x_t^{(3)}$, $x_t^{(5)}$ and $x_t^{(8)}$, and zero for $x_t^{(1)}$, $x_t^{(4)}$, $x_t^{(6)}$ and $x_t^{(7)}$. The first LLR vector can therefore be calculated as in (21). The second and third LLR vectors are calculated in the same fashion as in (22) and (23). The GET_LLR function calculates the three LLR vectors, after which these vectors are multiplexed in function MULTIPLEX. The LLR estimates are used in (20) during the next iteration.

Result The result of the DBN-TE algorithm is determined after the final iteration when $i = Z$, by transforming the LLR sequence corresponding to the first output bit of the encoder, into a bit sequence. The first LLR vector is used because the encoder in Figure 9 is systematic.

6 Complexity analysis

The computational complexity of the DBN-TE and the conventional turbo equalizer (CTE) are presented in this section. The complexity equations were derived by counting the number of computations needed to perform Turbo Equalization. The complexity of the DBN-TE was determined as

$$\mathrm{CC}_{\mathrm{DBN-TE}} = Z(2N_cM_d + 3N_cM_dQ/R_c), \tag{28}$$

where Z is the number of turbo iterations, M_d is the number of decoder states determined by 2^{K-1} where K is the encoder constraint length, Q is the number of interfering symbols which can be approximated by $Q \approx 2L - 1$, and R_c is the code rate. The approximation for Q was obtained empirically by calculating the average number of interfering symbols in the new channel $\mathbf{U}$ in (13) for a given original channel length L, after the transformation in (12) is applied. The complexity of the CTE was determined as

$$\mathrm{CC}_{\mathrm{CTE}} = Z(4N_cM_eL + 4N_uM_d/R_c), \tag{29}$$

where M_e is the number of equalizer states determined by 2^{L-1} for BPSK modulation.

Figure 13 shows the computational complexity graphs of the DBN-TE and the CTE, normalized by the number of coded transmitted symbols, for channel IR lengths from $L = 1$ to $L = 20$, for $Z = 3$ and $Z = 5$. $R_c = 1/3$, $N_u = 400$, $N_u = 1200$ and $K = 3$. From Figure 13 it can

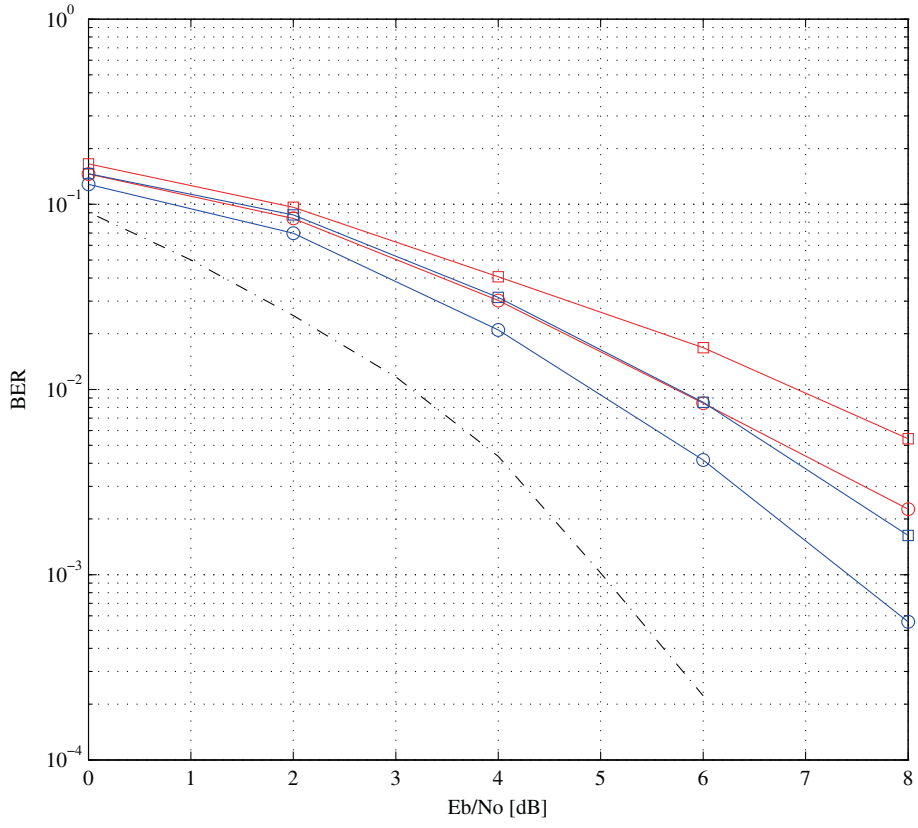

Figure 19 DBN-TE performance with and without simulated annealing. Red circle, CTE: 50 km/h; Red square, DBN-TE: 50 km/h; Blue circle, CTE: 20 km/h; Blue square: DBN-TE: 20 km/h; Black dashed: Decoded AWGN.

be seen that the computational complexity of the DBN-TE is much higher (10 times higher at $L = 2$) than that of the CTE for systems with channel IR lengths $L < 6$. However, as L increases beyond $L = 6$, the computational complexity of the DBN-TE becomes significantly less than that of the CTE. The DBN-TE is therefore a good candidate for systems with channel IR lengths $L > 6$. Note that for $L = 20$ the complexity of the DBN-TE is four orders less than that of the CTE.

7 Simulation results

The DBN-TE was evaluated in a mobile fading environment for BPSK modulation, where we used the Rayleigh fading simulator in [16] to generate uncorrelated fading vectors. Simulations were performed at varying mobile speeds and different channel IR lengths, where the energy in the channel was normalized such that $\mathbf{h}^{\dagger}\mathbf{h} = 1$. The channel IR was "estimated" by taking the mean of the respective fading vectors in order to get estimates for the channel IR coefficients, unless otherwise stated. The uncoded data block length was chosen to be $N_u = 400$ and the coded data block length was $N_c = 1200$, where the rate $R_c = 1/3$ convolutional encoder with generator polynomials $g_1(x) = 1$, $g_2(x) = 1 + x$, $g_3(x) = x + x^2$ in Figure 9 was used. Frequency hopping was also employed to reduce the BER.

Figures 14 and 15 show the performance of the DBN-TE and the CTE for different mobile speeds through channels with channel IR lengths of $L = 6$ and $L = 8$, respectively. The frequency was hopped eight times, once for every 150 transmitted symbols. The DBN-TE was simulated for $Z = 3$ iterations while the CTE was simulated for $Z = 5$ iterations. From Figures 14 and 15 it can be seen that the performance of the DBN-TE is less than a decibel worse than that of the CTE for mobile speeds of 3 and 20 km/h, while the performance of the DBN-TE closely matches the performance of the CTE for a mobile speed of 50 km/h. For mobile speeds of 80 and 110 km/h the DBN-TE outperforms the CTE. However, this result is of little practical importance, as the performance of both turbo equalizers at 80 and 110 km/h is no longer acceptable.

Figures 16 and 17 show the performance of the DBN-TE and the CTE for a fixed mobile speed of 20 km/h but with varying numbers of frequency hops, through channels with channel IR lengths of $L = 6$ and $L = 8$, respectively. From Figures 16 and 17 it can be seen

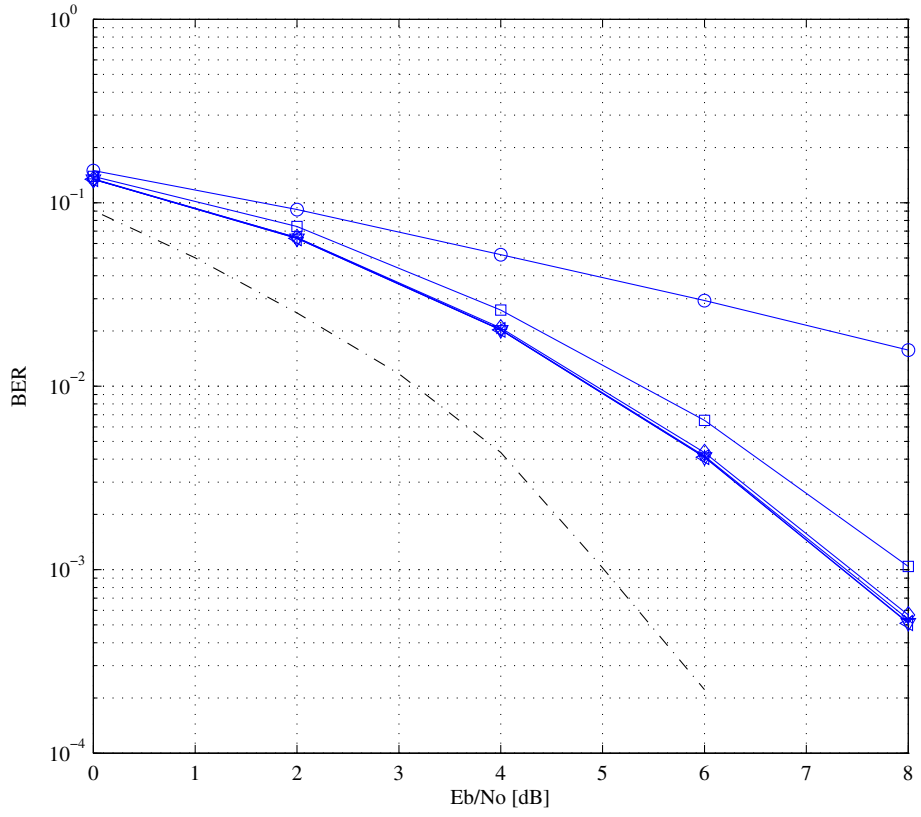

Figure 20 DBN-TE performance for different numbers of iterations. Blue circle, DBN-TE - 1 iteration; Blue square: DBN-TE - 2 iterations; Blue diamond: DBN-TE - 3 iterations; Blue star: DBN-TE - 4 iterations; Blue downward triangle: DBN-TE - 5 iterations; Blue leftward triangle: DBN-TE - 8 iterations; Black dashed: Decoded AWGN.

that the performance of the DBN-TE is again worse by less than a decibel for zero, two, four, and eight frequency hops.

Figure 18 shows the performance of the DBN-TE for channel IR lengths of $L = 5$, $L = 10$, and $L = 20$ at a mobile speed of 20 km/h using 8 frequency hops for $Z = 5$ iterations. From Figure 18 it is clear that the DBN-TE is able to turbo equalize signals in systems with longer memory, due to its low complexity. With reference to Figure 13, the number of computations required by the DBN-TE for $L = 10$ to $L = 20$ is in the range $10^{3.84} - 10^{4.15}$, whereas the number of computations required by the CTE is in the range $10^{5.01} - 10^{8.32}$.

In Section 5.2.7, optimization via simulated annealing was discussed. To demonstrate the effect of simulated annealing in the DBN-TE, simulations were performed with and without annealing for a channel IR length of $L = 6$ at speeds of 20 and 50 km/h, while the frequency was hopped four times (once for every 300 transmitted symbols) using $Z = 3$ iterations. Figure 19 shows the performance of the DBN-TE with and without simulated annealing. It is clear that the application of simulated annealing aids in improving the performance of the DBN-TE, where improvements of approximately 1 dB are achieved for the selected scenarios.

In order to demonstrate the speed of convergence of the DBN-TE, Figure 20 shows the performance of the DBN-TE for different numbers of iterations (Z), where the channel IR length is $L = 5$ at a mobile speed of 20 km/h using eight frequency hops. From Figure 20 it can be seen that there is no significant increase in performance for $Z > 3$. The DBN-TE therefore almost fully converges after only three iterations.

The simulation results in Figures 14, 15, 16, 17, 18, 19 and 20 were produced under the assumption that the channel state information (CSI) is known, or that at least the very best estimate thereof is available, due to the averaging of each uncorrelated fading vector as explained earlier. To evaluate the robustness of the DBN-TE with respect to CSI uncertainties, the channel was estimated with a least squares (LS) estimator, using various amounts of training symbols. Figure 21 shows the performance of the DBN-TE and the CTE for a channel IR length of $L = 6$ at a mobile speed of 20 km/h using 8 frequency hops, for $4L$, $6L$, and $8L$ training symbols in each frequency hopped segment of the transmitted data block. From Figure 21 it can be seen that the performance of

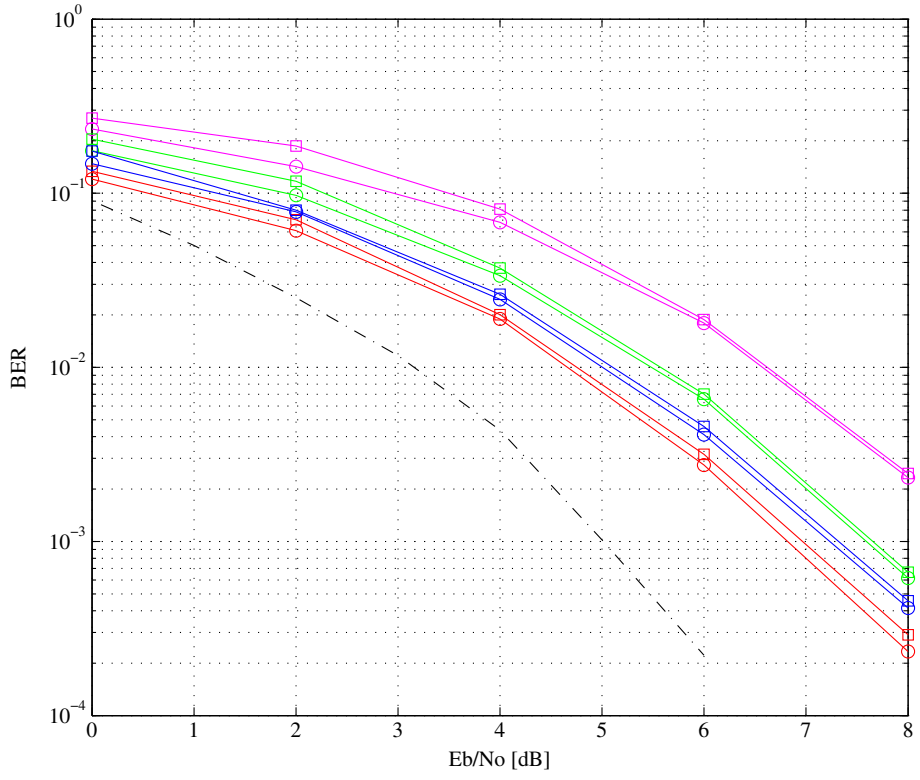

Figure 21 DBN-TE and CTE performance for various amounts of training symbols used in channel estimation. Red circle, CTE: Perfect CSI; Red square, DBN-TE: Perfect CSI; Blue circle, CTE: 8*L* pilots; Blue square, DBN-TE: 8*L* pilots; Green circle, CTE: 6*L* pilots; Green square, DBN-TE: 6*L* pilots; Pink circle, CTE: 4*L* pilots; Pink square, DBN-TE: 4*L* pilots; Black dashed, Decoded AWGN.

the DBN-TE degrades, along with that of the CTE, due to insufficient amounts of training symbols, causing an increase in channel estimation errors. Is clear that the DBN-TE is as resilient against channel estimation errors as the CTE, as its performance closely matches that of the CTE.

The results presented in this section are self-evident and show that the DBN-TE achieves acceptable performance compared to that of the CTE. There is a small degradation in BER performance compared to that of the CTE for all simulation scenarios, while a significant computational complexity reduction is achieved for systems with longer memory. The complexity of the DBN-TE is only linearly related to the number of channel IR coefficients, while the complexity of the CTE is exponentially related to the channel IR coefficients, allowing the DBN-TE to be applied to systems with longer channels. The fast convergence of the DBN-TE demonstrated in Figure 20 also adds to the complexity reduction, as full convergence is achieved after only three iterations.

8 Conclusion

In this article, we have proposed and motivated a turbo equalizer modeled on a DBN which uses belief propagation via the forward–backward algorithm, together with a soft-feedback mechanism, to jointly equalize and decode the received signal in order to estimate the transmitted symbols. We have motivated theoretically that this approach guarantees full convergence under certain conditions and we have shown that the performance of the new DBN-TE closely matches that of the CTE, with and without perfect CSI knowledge. Its complexity is linear in the coded data block length, exponential in the encoder constraint length, but only approximately linear in the channel memory length, which makes it an attractive alternative for use in systems with highly dispersive channels.

Additional file

Additional file 1: DBN-TE Pseudocode algorithm. (**a**) DBN-TE function pseudocode. (**b**) FORWARD_MESSAGE function pseudocode. (**c**) BACKWARD_MESSAGE function pseudocode. (**d**) FORWARD_BACKWARD_MESSAGE function pseudocode. (**e**) LLR_ESTIMATES function pseudocode.

Competing interests
The authors declare that they have no competing interests.

Author details
[1]Department of Electrical, Electronic and Computer Engineering, University of Pretoria, Pretoria 0002, South Africa. [2]School of Engineering, University of Tasmania, Hobart 7001, Australia. [3]Department of Mathematics and Applied Mathematics, University of Pretoria, Pretoria 0002, South Africa.

Received: 9 November 2011 Accepted: 23 May 2012
Published: 9 July 2012

References

1. C Berrou, A Glavieux, P Thitimajshima, in *IEEE International Conference on Communications*. Near Shannon limit error-correction and decoding: turbo-codes (1), vol 2, Geneva, 1993, pp. 1064–1070
2. C Douillard, M Jezequel, C Berrou, A Picart, P Didier, A Glavieux, Iterative correction of intersymbol intereference: turbo-equalization, Eur. Trans. Teleommun. **6**, 507–511 (1995)
3. G Bauch, H Khorram, J Hagenauer, in *Proceedings of European Personal Mobile Communications Conference (EPMCC)*. Iterative equalization and decoding in mobile communication systems, vol 2, 1997, pp. 307–312
4. R Koetter, M Tuchler, A Singer, Turbo equalization, IEEE Signal Process. Mag. **21**, 67–80 (2004)
5. R Koetter, M Tuchler, A Singer, Turbo equalization: principles and new results, IEEE Trans. Commun. **50**(5), 754–767 (2002)
6. R Lopes, J Barry, The soft feedback equalizer for turbo equalization of highly dispersive channels, IEEE Trans. Commun. **54**(5), 783–788 (2006)
7. D Mackay, *Information Theory, Inference, and Learning Algorithms*. (Cambridge University Press, Cambridge, 2003)
8. N Friedman, D Koller, *Probabilistic Graphical Models: Principles and Techniques*. (MIT Press, Cambridge, 2009)
9. Y Weiss, *Belief Propagation and Revision in Networks with Loops*. (MIT Press, Cambridge, 1997)
10. Y Weiss, W Freeman, Correctness of belief propagation in Gaussian graphical models of arbitrary topology, Neural. Comput. **13**, 2173-2200 (2001)
11. J Proakis, *Digital Communications*, 4th edn. (McGraw-Hill, International Edition, New York, 2001)
12. MI Jordan, KP Murphy, Y Weiss, in *Proceedings of Conference on Uncertainty in Artificial Intelligence*. Loopy belief propagation for approximate inference: an empirical study, vol 15, Stockholm, 1999, pp. 467–475
13. Y Weiss, Correctness of local probability propagation in graphical models with loops, Neural Comput. **12**, 1–41 (2000)
14. W Gerstacker, F Obernosterer, R Meyer, J Huber, On prefilter computation for reduced-state equalization, IEEE Trans. Wirel Commun. **1**(4), 793–800 (2002)
15. S Russell, P Norvig, *Artificial Intelligence: A Modern Approach*, 2nd edn. (Prentic Hall, NJ, 2003)
16. Y Zheng, C Xiao, "Improved models for the generation of multiple uncorrelated Rayleigh fading waveforms", IEEE Commun Lett. **6**, 256–258 (2002)

doi:10.1186/1687-6180-2012-136
Cite this article as: Myburgh *et al.*: Reduced complexity turbo equalization using a dynamic Bayesian network. *EURASIP Journal on Advances in Signal Processing* 2012 **2012**:136.