

Generalized Versions of Turbo Decoding in the Framework of Bayesian Networks and Pearl's Belief Propagation Algorithm

Peyman Meshkat and John D. Villasenor

Electrical Engineering Department
University of California, Los Angeles

Abstract— We use the framework of Bayesian networks to introduce generalizations of the traditional turbo decoding algorithm. We show that traditional turbo decoding represents one of many ways in which the framework of Pearl's belief propagation algorithm can be applied for decoding of turbo codes. Simulation results show that a noisy received block which does not converge using traditional turbo decoding can converge to the correct value with one or more of the generalizations introduced here. Though we consider only the case of turbo codes with two constituent decoders here, these methods can be extended in a straightforward manner to codes with larger numbers of constituent decoders.

I. INTRODUCTION

Since the introduction of turbo codes in 1993 by Berrou, Clavier and Thitimajshima, [1] many researchers have worked to analyze and improve turbo code performance by considering aspects such as choice of generating polynomial, feed forward vs. feedback, number of states, and interleaver depth [2], [4], [3], [5]. Significantly less work has been done on the convergence behavior of turbo decoding, and essential questions remain regarding the probability of converging to a non-optimal decision, the probability and conditions under which the decoding iterations will result in a non-converging sequence of decisions, and whether non-converging blocks have limit-cycle behavior. Such questions will likely be answered only when a formal description of turbo codes can be constructed and mathematically analyzed, and as the authors of publications including [6], [7], [8] have observed, graphical representations have the potential to supply such a description.

Within graphical representations, Bayesian networks and Pearl's belief propagation algorithm [9] show particular promise for application to turbo decoding. Pearl's belief propagation algorithm is a message propagation algorithm for calculating *a-posteriori* probabilities for a set of random variables which construct a Bayesian network. The algorithm is quite well known in the artificial intelligence (AI) community, and has recently begun to receive attention in the coding theory community as well. For example, the close connections between turbo decoding algorithm and Pearl's belief propagation algorithm have been described by MacKay, McEliece and Cheng in [7] and by Kschischang and Frey in [8]. Pearl's algorithm calculates the exact *a-posteriori* probabilities (or "belief" in the AI literature) of the variables for a loop-free Bayesian network. Although the Bayesian network corresponding to turbo codes is not loop-free, MacKay *et al.* have shown in [7] that turbo decoding can nonetheless be viewed as an implementation of Pearl's belief propagation algorithm in which the presence of loops is ignored.

The order in which nodes are processed in a network is re-

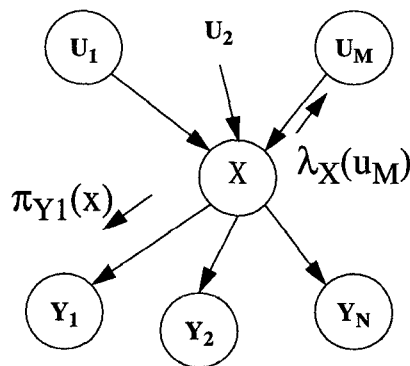


Fig. 1. A fragment of a Bayesian network. Node X sends λ messages to its parents (U nodes) and π messages to its children (Y nodes).

ferred to as the "node activation schedule" or, equivalently, the "timing schedule." In a loop-free network all different timing schedules result in a unique and exact solution. On the graph representing turbo decoding, however, an exact solution is no longer guaranteed, and we show here that the outcome becomes a function of the timing schedule that is used to activate the nodes in the graph. This opens the possibility of exploring the relationship between timing schedule and decoding performance. We show that alternative timing schedules can in fact allow performance approximately equal to that achieved by traditional turbo decoding decoding based on the forward/backward algorithm of Bahl *et al.* [12]. We emphasize that these alternative timing schedules correspond to different decoding algorithms, and while the average performance is equivalent to that of the traditional forward/backward algorithm, the performance on individual blocks often differs. Our use of the term "traditional" turbo decoding is not meant to imply that there is only one way that turbo codes have been implemented. However, almost all decoding approaches presented to date use the forward/backward algorithm to perform processing in one constituent decoder at a time during each pass through the decoding iterations, with information exchange to the other constituent decoder(s) occurring via the interleaver only after all nodes in one of the trellises have been considered.

By contrast, the framework we adopt constitutes a more general approach that allows the nodes of the graph representing turbo decoding to be activated in an arbitrary order. This opens the possibility of processing some nodes in a second constituent decoder before all of the nodes in the first constituent decoder have been considered, and of performing the message passing between decoders via the interleaver in a similarly distributed manner. The only other study to con-

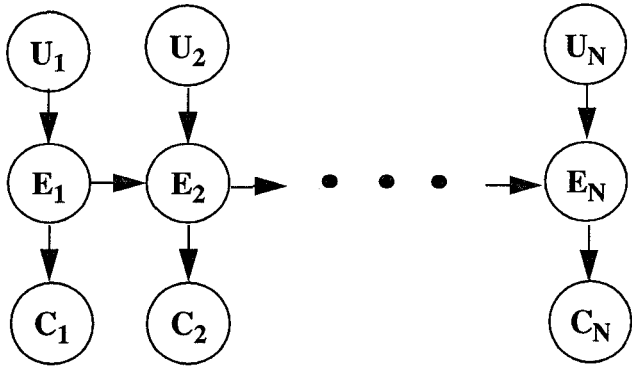


Fig. 2. Bayesian network corresponding to a convolutional code. U nodes represent information bits, E nodes represent transition edges and C nodes represent channel outputs. The block size is N , with elements enumerated by the subscripts of U , E , and C .

consider different timings for message propagation in the context of turbo codes is provided by Frey *et al.* [10], who introduce a concurrent turbo decoding algorithm and explore the effect of activating all nodes simultaneously on computational complexity and convergence. We consider only serial timing schedules here (though the framework admits concurrent schedules such as that in [10] as well), so the complexity required is identical to that found in traditional turbo coding since each node is activated once per decoding iteration. In addition to showing that turbo codes can be successfully decoded using the many different algorithms that correspond to alternative schedules, we show that there are some “problematic” blocks which do not converge when decoded using traditional turbo decoding, but which do converge when one or more alternative timing schedules are used. This result is promising in that it offers proof that a more general approach to timing schedules can, at least for some cases, improve the performance of turbo codes. In the following sections, we provide a very brief overview of belief propagation, and then show how it applies to turbo decoding. We then present several node activation schedules and give simulation results that show their decoding performance.

II. THE BAYESIAN NETWORK CORRESPONDING TO TURBO CODES

A Bayesian network consists of a set of random variables $X = \{X_1, X_2, \dots, X_N\}$ which are represented by the nodes of the network and by directed links from parent nodes to their children. To construct a Bayesian network, one first assigns some ordering to the variables $\{X_1, X_2, \dots, X_N\}$, and then, starting from the first variable, continues to add the new variables (nodes) to the graph. Once a new node X_k has been added to the network, new links from a subset $Pa(X_k)$ of the previous nodes $\{X_1, X_2, \dots, X_{k-1}\}$ to the newly added node X_k are added to the network. $Pa(X_k)$ is called the parent set of the node X_k and is a minimal subset of the previous nodes (i.e. given $Pa(X_k)$, the random variable X_k is independent of all previous nodes, or more precisely, of all random variables corresponding to those nodes). This construction corresponds to decomposing the joint probability

distribution function using the chain rule but with having a minimal set of variables after the condition bar. A Markov chain can be considered as one of the simplest examples of a Bayesian network. For example, in a Markov chain, given the present random variable the previous and next (parents and children) random variables are independent of each other. The graph corresponding to general a Bayesian network may contain loops.

Pearl’s belief propagation algorithm is a message passing algorithm for calculating *a-posteriori* probabilities of nodes of a loop-free Bayesian network given *a-priori* probabilities and some observation. A detailed description of belief is provided by Pearl in [9]. Here we describe only the basic concepts that are necessary for presenting the ideas introduced in this work. Pearl’s algorithm consists of activation of the nodes as well as the timing schedule determining the order in which activation should occur. Activation of a node X essentially involves updating the information associated with the node based on the information available to it from its neighbors. This is formally specified by the three steps given below and illustrated in Figure 1. Note that nodes U_k represent parents of X and nodes Y_k represent their children.

Step 1. Belief updating: Calculation of the *a-posteriori* probabilities of the random variable corresponding to the node X , given the information about the evidence that comes from child nodes (λ messages) and information which comes from the parent nodes (π messages).

$$BEL(x) = \alpha \lambda(x) \pi(x) \quad (1)$$

where

$$\lambda(x) = \prod_j \lambda_{Y_j}(x), \quad (2)$$

$$\pi(x) = \alpha \sum_{u_1, \dots, u_n} P(x|u_1, \dots, u_n) \prod_i \pi_X(u_i), \quad (3)$$

and the *a-posteriori* probability or “Belief”, as used in AI literature, is denoted by BEL.

Step 2. Top down propagation: Computation of the *a-posteriori* probability of the node given all evidence except for information that comes from a given child node. This probability would be the parent to child message for the child node whose information is excluded. The message from parent node X to the child node Y_j is denoted by $\pi_{Y_j}(x)$.

$$\pi_{Y_j}(x) = \alpha \left[\prod_{k \neq j} \lambda_{Y_k}(x) \right] \sum_{u_1, \dots, u_n} P(x|u_1, \dots, u_n) \prod_i \pi_X(u_i) \quad (4)$$

Step 3. Bottom up propagation: Computation of the conditional probability of the evidence coming from the children of U given different possible values for the random variable corresponding to node U . The message from child node X to the parent node U_i is denoted by $\lambda_X(u_i)$.

$$\lambda_X(u_i) = \beta \sum_x \lambda(x) \sum_{u_k; k \neq i} P(x|u_1, \dots, u_n) \prod_{k \neq i} \pi_X(u_k) \quad (5)$$

A derivation of the above and treatment of boundary values for λ and π messages is given in [9].

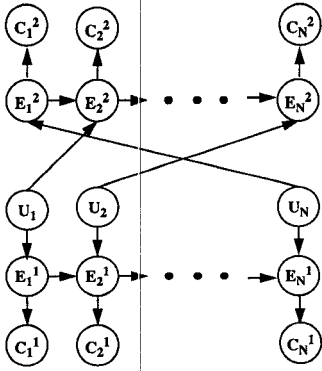


Fig. 3. Bayesian network corresponding to a turbo code with two constituent codes. As in Figure 2, U nodes represent information bits, E nodes represent transition edges and C nodes represent channel outputs. Superscripts identify the constituent code. The interleaver acts between the information bits U and the second constituent coder E^2 .

The Bayesian network corresponding to a single convolutional code as shown in Figure 2. The variables E_k , $k = 1, 2, \dots, N$ correspond to the k th trellis transition edge where N is the size of the block. A trellis edge is defined by state and input which are considered jointly as one random variable. If state and input symbol were considered separately the corresponding graph would be loopy one as noted in [8]. Figure 3 shows the graph corresponding to a parallel concatenated convolutional (turbo) decoder with two constituent decoders. The nodes $\{E_1^1, \dots, E_N^1\}$ represent trellis transitions of the first decoder and nodes $\{E_1^2, \dots, E_N^2\}$ represent trellis transitions in the second decoder. The information bits are represented by the letter U (for uncoded). The observations (i.e. the noise-corrupted outputs) from the first and second constituent coders are denoted by $C^1 = \{C_1^1, \dots, C_N^1\}$ and $C^2 = \{C_1^2, \dots, C_N^2\}$ respectively. The systematic bits are considered as part of the output C^1 of the first constituent coder.

III. GENERALIZED TURBO DECODING

Given the graph of Figure 3, there are clearly many different node activation schedules that can be used. Below we introduce four such schedules and explore their performance via simulation. In each decoding iteration, it is assumed that all of the nodes corresponding to noisy observations (all C nodes in Figure 3) have been activated before starting processing using the activation schedules. Since the C nodes only produce child to parent (λ) messages the order of activation of these nodes does not effect the final result, and we need only consider different activation schedules for the nodes corresponding to trellis edges (E nodes) and information bits (U nodes).

Method 1: This method corresponds to traditional turbo decoding. We consider E^1 and E^2 as two sets of random variables whose elements are denoted by $E^1 = \{E_1^1, \dots, E_N^1\}$ and $E^2 = \{E_1^2, \dots, E_N^2\}$ respectively, where each element E_j^i is the j th trellis transition edge in the i th constituent decoder. Using this notation, activation of E^1 corresponds to activation of the nodes $E_1^1, E_2^1, \dots, E_N^1$ first

	Node activation schedule			
	Method 1	Method 2	Method 3	Method 4
residual BER	0.0099	0.0097	0.0097	0.0094

Tab. I: Residual BER for different node activation schedules. The channel is AWGN with an SNR of .3 dB. Method 1 corresponds to traditional turbo decoding using the forward/backward algorithm. Methods 2-4 represent alternative schedules as described in the text. The BERs for all methods are approximately equivalent, showing that the traditional approach using the forward/backward algorithm (Method 1) is just one of many ways in which approximately equivalent code performance can be achieved.

in forward direction and then in the backward direction. This is equivalent to the forward/backward algorithm [12]. If we denote ACTIV as a subroutine that performs the three steps mentioned in Section 2 this method corresponds to the following:

for $i = 1 : N$
ACTIV(E_i^1)

for $i = N : 1$
ACTIV(E_i^1)

for $i = 1 : N$
ACTIV(E_i^2)

for $i = N : 1$
ACTIV(E_i^2)

Method 2: One can also activate the nodes corresponding to the two constituent decoders in alternation, while preserving the forward/backward structure of the activation as follows:

for $i = 1 : N$
ACTIV(E_i^1)
ACTIV(E_i^2)

for $i = N : 1$
ACTIV(E_i^1)
ACTIV(E_i^2)

The key difference between this method and traditional turbo decoding (method 1) is that here the two constituent coders are considered together. This type of activation schedule could also be applied to turbo codes with three or more constituent coders such as those described in [11].

Method 3: This method is similar to method 1 but with the key difference that immediately after activation of each node in the first constituent decoder, the node in the second constituent decoder corresponding to the same input bit is activated. To express this we use $int(i)$ to denote the position to which i th input bit is mapped by the interleaver. The inverse $int^{-1}(i)$ identifies the position of the input bit which is mapped to location i by the interleaver.

$for\ i = 1 : N$
 $ACTIV(E_i^1)$
 $ACTIV(E_{int(i)}^2)$

$for\ i = N : 1$
 $ACTIV(E_i^1)$
 $ACTIV(E_{int(i)}^2)$

$for\ i = 1 : N$
 $ACTIV(E_i^2)$
 $ACTIV(E_{int^{-1}(i)}^1)$

$for\ i = N : 1$
 $ACTIV(E_i^2)$
 $ACTIV(E_{int^{-1}(i)}^1)$

Method 4: This method combines the approach of activating nodes in both decoders in alternation (from Method 2) with the explicit use of the interleaver structure (from Method 3).

$for\ i = 1 : N$
 $ACTIV(E_i^1)$
 $ACTIV(E_{int(i)}^2)$
 $ACTIV(E_i^2)$
 $ACTIV(E_{int^{-1}(i)}^1)$

$for\ i = N : 1$
 $ACTIV(E_i^1)$
 $ACTIV(E_{int(i)}^2)$
 $ACTIV(E_i^2)$
 $ACTIV(E_{int^{-1}(i)}^1)$

We have implemented the activation schedules above using a rate 1/3 encoder that included a systematic bit and two identical recursive 8-state convolutional encoders with generator matrix $G(D) = \frac{1+D+D^2+D^3}{1+D^2+D^3}$ and an interleaver with length 1024. Each simulation consisted of at least 25 million bits (approximately 25000 blocks). In order to ensure that a statistically significant number of blocks would fail to converge we chose a low signal to noise ratio of 0.3 dB.

Table 1 shows the residual bit error rates (BER) corresponding to methods 1 through 4, and shows that these methods yield almost identical BER performance. However this does not mean that the blocks for which decoding is unsuccessful are the same in all methods. Table 2 shows the number of blocks (both as an absolute number and as a percentage) which were correctly decoded using Methods 2-4, given that they did not converge to the correct solution in a turbo decoder based on the forward backward algorithm (Method 1). These data are based on a total of 269 blocks which failed to correctly converge for Method 1. They show that while most blocks that are non-convergent for Method 1 remain non-convergent for Methods 2-4, some of the blocks

	Node activation schedule		
	Method 2	Method 3	Method 4
Number of correctly decoded "probematic" blocks	15	10	17
convergence percentage	5.6	3.7	6.3

Tab. II: **Convergence percentage for problematic blocks.** Number of blocks which were correctly decoded using Methods 2-4, given that they did not converge to the correct solution in a turbo decoder based on the forward backward algorithm (Method 1). The second row of the table gives the equivalent percentage. These data are based on a total of 269 blocks which failed to converge for Method 1. They show that while most blocks that are non-convergent for Method 1 remain non-convergent for Methods 2-4, some of the blocks that are non-convergent for Method 1 can be correctly decoded when other node activation schedules are used.

that are non-convergent for Method 1 can be correctly decoded when other node activation schedules are used.

IV. CONCLUSIONS

We have shown that that the traditional turbo decoding algorithm using the forward/backward algorithm is not the only iterative way of decoding parallel concatenated convolutional codes. The framework of Bayesian networks, with the aid of Pearl's belief propagation algorithm, allows other possible orderings for activation of the nodes. Simulation results show that all of these methods lead to approximately the same residual BER. This supports the intuition that it is the long block length which is largely responsible for the performance of turbo codes, and that there are many decoding algorithms which can be used to process these large blocks. While the activation schedules we present yield equivalent performance in the average sense, their behavior on individual blocks can be quite different. We have demonstrated that there are blocks for which traditional turbo decoding using the forward/backward algorithm fails but the alternative activation schedules give error-free decoding. This suggests that decoding techniques that involve simultaneous processing using several activation schedules or on-the-fly adaptation of the activation schedule hold promise. Although we have considered only the case of two constituent decoders here, it is also possible to apply these techniques to codes with > 2 constituent coders.

REFERENCES

- [1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo Codes," *Proc. of ICC '93*, pp. 1064-1070, 1993.
- [2] S. Benedetto, G. Montorsi, D. Divsalar and F. Pollara, "Serial concatenation of interleaved codes: performance analysis, design, and iterative decoding," *JPL TDA Progress Report*, vol. 42-126 (Aug. 1996), in press.
- [3] J. Hagenauer, E. Offer, and L. Papke, "Iterative Decoding of Binary Block and Convolutional Codes," *IEEE Trans. on Information Theory*, pp. 429-445, March 1996.
- [4] S. Benedetto, G. Montorsi, "Unveiling Turbo Codes: Some Results on Parallel Concatenated Coding Schemes," *IEEE Trans. on Information Theory*, pp. 409-428, March 1996.
- [5] L. C. Perez, J. Seghers, and D. Costello, "A Distance Spectrum Interpretation of Turbo Codes," *IEEE Trans. on Information Theory*, pp. 1698-1709, November 1996.

- [6] N. Wiberg, "Codes and Decoding on General Graphs", Linkoping Studies in Science and Technology, Dissertation No. 440. Linkoping, Sweden, 1996.
- [7] D.J.C. MacKay, R.J. McEliece and J.F. Cheng, "Turbo Decoding as an Instance of Pearl's Belief Propagation Algorithm", Submitted to *IEEE Journal on Selected Areas in Communication*, Draft of November 14, 1996.
- [8] F. Kschischang and B. Frey, "Iterative decoding of compound codes by probability propagation in graphical models", submitted to *IEEE Journal on Selected Areas in Communication*, Sept. 1996.
- [9] J. Pearl, "Probabilistic Reasoning in Intelligent Systems", San Mateo, CA: Morgan Kaufmann, 1988.
- [10] B.J. Frey, F.R. Kschischang and P. Blenn Gulak, "Concurrent Turbo-Decoding", *Proc. of ISIT '97*, pp. 431, 1997.
- [11] D. Divsalar, F. Pollara, "Turbo Codes for PCS Applications," *Proc. of the International Conference in Communications*, pp. 54-59, 1995.
- [12] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate," *IEEE Trans. on Inf. Theory*, pp. 284-287, March 1974.