# OPTIMUM NON-ITERATIVE TURBO-DECODING

*M. Breiling, L. Hanzo*

Dept. of Electr. and Comp. Sc., Univ. of Southampton, SO17 1BJ, UK.
Tel: +44-703-593 125, Fax: +44-703-594 508
Email: lh@ecs.soton.ac.uk
http://www-mobile.ecs.soton.ac.uk

## ABSTRACT

**By observing the structure of the decoder's trellis a new, non-iterative turbo-decoder based on a super-trellis structure is proposed, which exhibits the same decoding complexity as a conventional convolutional decoder posessing an identical number of trellis states. For the investigated half-rate, memory-length two code the proposed algorithm requires about 0.5 dB lower Gaussian channel signal-to-noise ratio (SNR) than the Maximum A Posteriori (MAP) algorithm using 16 iterations.**

## 1. INTRODUCTION

Turbo coding was originally proposed by Berrou, Glavieux and Thitimajshima [2] in order to achieve near-Shannonian performance over Gaussion channels. Hagenauer and Hoeher proposed to use the soft-output Viterbi algorithm for the decoding of Turbo codes in Reference [3] , while Hagenauer, Offer and Papke [4] investigated also the feasibility of employing block codes as constituent codes, although here we will concentrate on convolutional codes. Various turbo decoders were investigated by Robertson et al [5] and Jung [6].

The outline of the paper is as follows. Section 2 presents the basic decoding philosophy in the context of conventional convolution decoding, while Section 3 concentrates on the proposed decoding technique, leading to Section 4 presenting our simulation results.

## 2. DECODING CONVOLUTIONAL CODES

In order to introduce our formalism, Figure 1 shows an example of a path through the trellis for a codeword $\mathbf{c}_i$, where the quantities $\bar{c}_{i,j}$ along the path represent the symbol sequence within the codeword $\mathbf{c}_i$ that is associated with the trellis state transition $j$ and the corresponding encoder input bit $u_j$. For finding the most likely transmitted codeword, we define the following path metrics (PM):

$$\mathbf{M}_{\mathbf{c}_i,j\leq k} := \sum_{j=1}^{k} \|\bar{c}_{i,j} - \bar{r}_j\|^2 \hat{=} \text{forward PM} \qquad (1)$$

$$\mathbf{M}_{\mathbf{c}_i,l<j} := \sum_{j=l+1}^{N} \|\bar{c}_{i,j} - \bar{r}_j\|^2 \hat{=} \text{backward PM.} \qquad (2)$$

These so-called path-metrics are constituted by the sum of consecutive branch-metrics. Each branch-metric quantifies the similarity or dissimilarity between the received sequence $\bar{r}_j$ and the codeword $\bar{c}_{i,j}$ at instant $j$. When considering the trellis stage $j$, the two associated trellis paths depicted in Figure 1a will be referred to as the forward path and the backward path, respectively. The parameter $N$ in Equation 2 is the length of the input dataword, in other words the total number of transitions in the trellis, while the $\bar{r}_j$ is the symbol sequence that has actually been received at stage $j$.

The terminology 'forward' and 'backward' path were chosen, because their metrics can easily be calculated by a forward/backward recursion as follows:

$$\mathbf{M}_{\mathbf{c}_i,j\leq k} = \mathbf{M}_{\mathbf{c}_i,j\leq k-1} + \|\bar{c}_{i,k} - \bar{r}_k\|^2 \qquad (3)$$

$$\mathbf{M}_{\mathbf{c}_i,l-1<j} = \mathbf{M}_{\mathbf{c}_i,l<j} + \|\bar{c}_{i,l} - \bar{r}_l\|^2. \qquad (4)$$

As we can see in Figure 1a), any codeword $\mathbf{c}_i$ can now be broken up into a forward path ending at trellis state transition $k$ and a backward path from this transition until the end of the trellis. Its total metric evaluated by the decoder consists therefore of two terms:

$$\mathbf{M}_{\mathbf{c}_i} = \mathbf{M}_{\mathbf{c}_i,j\leq k} + \mathbf{M}_{\mathbf{c}_i,k<j}\,. \qquad (5)$$

We make the following observation. If two codewords $\mathbf{c}_a$ and $\mathbf{c}_b$ differ only in terms of their forward paths with respect to the trellis stage $k$, while their backward paths are identical, then the codeword associated with the higher forward path metric can be discarded, because its total metric $\mathbf{M}_{\mathbf{c}_i}$ is greater than that of the other one and it can thus never be the minimum metric path. Two partial paths are identical, if they both commence and terminate in the same encoder state and are associated with the same data input bits along their way through the code trellis.

Moving on to the decoding process, this means that for any trellis stage $k$ (i.e. the $k^{th}$ transition in the trellis), we have to look at each of the $2^{K-1}$ possible states in the trellis (where $K$ is the constraint length of the encoder) and keep only the specific forward path with the minimum metric merging into this state. All other forward paths merging into the same state can be discarded. Then we can go on to the next decoding stage $k+1$, extend the surviving forward paths of each of the $2^{K-1}$ states at stage $k$ by one trellis transition, compute their metrics by a simple forward

The financial support of the following organisations is gratefully acknowledged: Motorola ECID, Swindon, UK; European Community, Brussels, Belgium; Engineering and Physical Sciences Research Council, Swindon, UK; Mobile Virtual Centre of Excellence, UK.

PIMRC'97, Sept. 1997, Helsinki, Finland
Session: Error Correction



0-7803-3871-5/97/$10.00 © 1997 IEEE

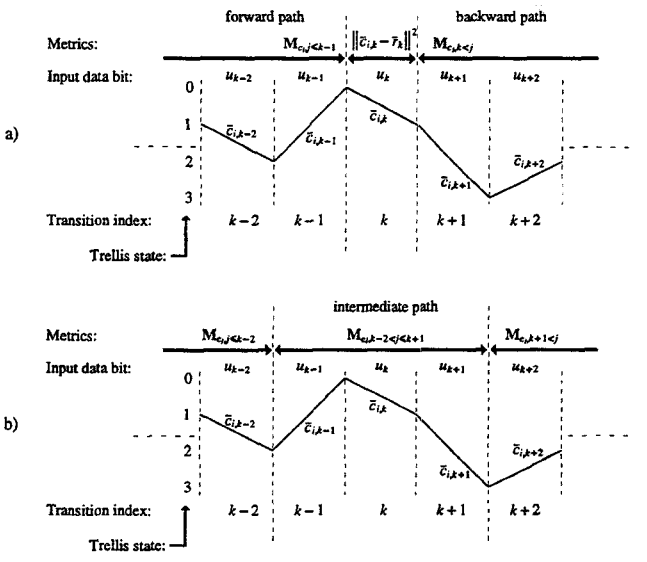


Figure 1: Examples for partial paths and their metrics

recursion, select for each state the forward path with the lowest metric etc.

We shall now introduce a new type of paths, which we refer to as the intermediate paths. This is a trellis segment associated with an intermediate section of a codeword as can be seen in Figure 1b and its metric is accordingly given by:

$$\mathbf{M}_{\mathbf{c}_i,k<j\leq l} := \sum_{j=k+1}^{l} \|\bar{c}_{i,j} - \bar{r}_j\|^2. \qquad (6)$$

With this notation, a codeword metric can be split up into three parts:

$$\mathbf{M}_{\mathbf{c}_i} = \mathbf{M}_{\mathbf{c}_i,j\leq k} + \mathbf{M}_{\mathbf{c}_i,k<j\leq l} + \mathbf{M}_{\mathbf{c}_i,l<j}\,. \qquad (7)$$

We could therefore set up a dynamic programming approach as follows. If two codewords $\mathbf{c}_a$ and $\mathbf{c}_b$ differ only in an intermediate path, we may argue that the codeword with the higher intermediate path metric $\mathbf{M}_{\mathbf{c}_i,k<j\leq l}$ can be discarded, before we go on to the next decoding stage. The reasoning for this is that their forward metric with respect to the $k^{th}$ symbol sequence and their backward metric with respect to the $l^{th}$ symbol sequence are identical, because their forward and backward paths are identical on these trellis segments. As discussed above for forward paths, the codeword with the higher intermediate metric has the higher total metric and can therefore be discarded. The surviving intermediate path could then be extended in either direction to explore the trellis in order to find the most likely path.

This shows that the Viterbi decoding process could also start in the middle of the trellis. But since 'identical apart from an intermediate path' now means, that the forward paths have to merge into the same state at stage $k$ and the backward paths have to start in the same state at stage $l$, we have to take into account every possible combination of states the intermediate path commences and terminates in. The number of states to take into account by this kind of algorithm is therefore squared in comparison to extending either forward or backward paths only, as in conventional Viterbi decoding.

## 3. DECODING TURBO CODES

Having explained the dynamic programming method for decoding convolutional codes, we are able to proceed to the more complex task of decoding turbo codes. We are going to highlight, why conventional turbo Decoders use an iterative method and how we can define an optimum non-iterative decoder.

An important difference between conventional convolutional codes and turbo codes is that the decoding process of the latter is not sequential. The effect of changing a symbol in one part of the codeword will not only affect possible paths in this part, but also the paths in distant parts of the codeword. In order to visualize this, the simplified encoder-/decoder structure is displayed in Figure 2, where we use the following notation:

- $\mathbf{u} = (u_k)_{k=1..N}$ represents the original non-interleaved data bit sequence, which is used as input for the first encoder
- $\mathbf{u}^{(2)} = (u_k^{(2)})_{k=1..N}$ is the interleaved bit sequence, which is used as input for the second encoder
- $\mathbf{x} = \mathrm{enc}(\mathbf{u})$ is the output sequence of the first encoder and
- $\mathbf{x}^{(2)} = \mathrm{enc}(\mathbf{u}^{(2)})$ is the ouput of the second decoder, where enc() denotes the encoding function
- $\mathbf{y}$ is the part of the received sequence belonging to $\mathbf{u}$, i.e. to the input sequence of the first decoder
- $\mathbf{y}^{(2)}$ is the part of the received sequence belonging to $\mathbf{u}^{(2)}$

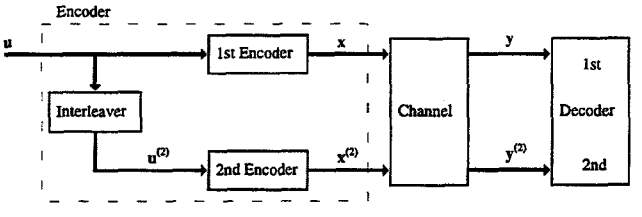


Figure 2: The turbo encoder/decoder structure

Figure 3 shows an example for the positions of the first seven bits of the input data sequence in $\mathbf{u}$ and $\mathbf{u}^{(2)}$, for the sake of illustration assuming a very simple interleaver algorithm, which becomes explicit from the Figure.

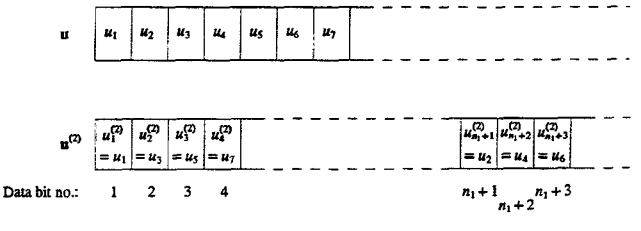


Figure 3: The original and the interleaved bit sequence

Each codeword $\mathbf{c}_i$ is made up of its two parts $\mathbf{x}_i$ and $\mathbf{x}_i^{(2)}$, and therefore we can refer to $\mathbf{x}$ and $\mathbf{x}^{(2)}$ as the partial codewords.

Figure 4 shows the stylised trellises corresponding to the interleaver of Figure 3 that are used to produce $\mathbf{x}$ and

$\mathbf{x}^{(2)}$. Explicitly, we have to consider not one trellis but two, and we have hence to introduce the following partial path metrics:

$$\mathbf{M}_{\mathbf{x}_i,j\leq k} := \sum_{j=1}^{k} \|\bar{x}_{i,j} - \bar{y}_j\|^2 \quad (8)$$

$$\mathbf{M}_{\mathbf{x}_i,l<j} := \sum_{j=l+1}^{N} \|\bar{x}_{i,j} - \bar{y}_j\|^2 \quad (9)$$

$$\mathbf{M}_{\mathbf{x}_i^{(2)},m\leq n} := \sum_{m=1}^{n} \left\|\bar{x}_{i,m}^{(2)} - \bar{y}_m^{(2)}\right\|^2 \quad (10)$$

$$\mathbf{M}_{\mathbf{x}_i^{(2)},n<m\leq o} := \sum_{m=n+1}^{o} \left\|\bar{x}_{i,m}^{(2)} - \bar{y}_m^{(2)}\right\|^2 \quad (11)$$

$$\mathbf{M}_{\mathbf{x}_i^{(2)},o<m} := \sum_{m=o+1}^{N} \left\|\bar{x}_{\mathbf{x}_i,m}^{(2)} - \bar{y}_m^{(2)}\right\|^2, \quad (12)$$

where $\bar{x}_{i,j}$ is the symbol sequence belonging to the $j^{th}$ trellis stage in the partial codeword $\mathbf{x}_i$, and $\bar{y}_j$ is the segment of the received sequence belonging to the $j^{th}$ trellis stage. Equations 8 and 9 define the forward/backward metric for the partial codeword $\mathbf{x}_i$, while Equations 10 to 12 define the same metrics for $\mathbf{x}_i^{(2)}$. Their definition is analogous to Equations 1, 2 and 6 and they are visiualized by the corresponding partial paths in Figure 1.

Since we are using the Euclidian distance for the metric $\mathbf{M}_{\mathbf{c}_i}$ of the complete turbo codeword $\mathbf{c}_i$, it is easy to show that the turbo-decoded metric is given by the sum of the constituent metrics:

$$\mathbf{M}_{\mathbf{c}_i} = \mathbf{M}_{\mathbf{x}_i} + \mathbf{M}_{\mathbf{x}_i^{(2)}}, \quad (13)$$

where $\mathbf{M}_{\mathbf{x}_i}$ and $\mathbf{M}_{\mathbf{x}_i^{(2)}}$ each can be broken up into three parts according to Equations 8 to 12 and Equation 7.

If we now attempt to use a dynamic programming approach for decoding $\mathbf{y}$ of Figure 2 and ignore $\mathbf{y}^{(2)}$, we start discarding forward paths in the upper trellis of Figure 4 while retaining the survivors. As shown in Section 2 in the context of decoding conventional convolutional codes, this way we are able to find the partial codeword $\mathbf{x}_i$ with the minimum metric $\mathbf{M}_{\mathbf{x}_i}$.

It would be very convenient, if we were able to consider the decoding of the lower trellis having found the optimum sequence in the upper trellis. However, this is not possible, since having decided for $\mathbf{x}_i$ as the most probable partial codeword in the upper trellis, also the complete codeword $\mathbf{c}_i$ and the other partial codeword $\mathbf{x}_i^{(2)}$ are determined as there is a unique relationship between these three quantities, and hence there is only one possible path left in the lower trellis.

The optimal path in the upper trellis does not have to be associated with the most likely path exhibiting the lowest metric $\mathbf{M}_{\mathbf{x}_i^{(2)}}$ in the lower trellis. By minimizing $\mathbf{M}_{\mathbf{x}_i}$, we do not necessarily minimize $\mathbf{M}_{\mathbf{c}_i}$, as other codewords $\mathbf{c}_j$ might have slightly greater metrics $\mathbf{M}_{\mathbf{x}_j}$, but much smaller metrics $\mathbf{M}_{\mathbf{x}_j^{(2)}}$, resulting in a smaller overall metric $\mathbf{M}_{\mathbf{c}_j}$. Due to the random nature of the channel outputs $\mathbf{y}$ and $\mathbf{y}^{(2)}$, it would be easy to find such an example.

Following the above arguments, we conclude that decoding a parallel concatenated convolutional code cannot be achieved by serially decoding its constituent trellises with a standard dynamic programming approach.

Berrou et al [2] proposed a solution to this problem by refraining from employing dynamic programming. Explicitly, instead of discarding potentially possible paths while identifying the most likely path, state-of-the-art techniques attempt to calculate the likelihood of each bit of the original dataword $\mathbf{u}$ of being 0 or 1 according to the first code trellis and the received sequence $\mathbf{y}$, and then pass this information on to the second decoder. The latter one uses this additional soft-decision information to recalculate the likelihood of the data sequence bits, but now according to the received sequence $\mathbf{y}^{(2)}$, and passes the new soft-decoded information back to the first decoder. Several of these iterations can be performed, before the soft-decoded information is used to produce a hard-decision decoder output. This

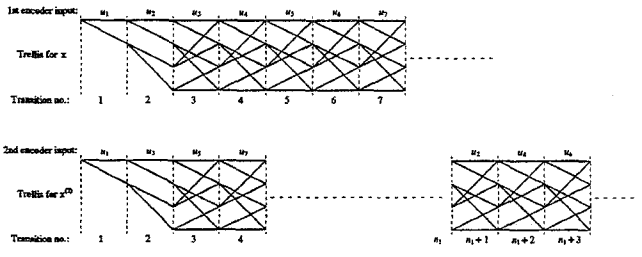

Figure 4: An example for the two encoder trellises

approach attempts to find the optimum dataword with the highest probability iteratively. The convergence speed varies and the computational power required to approach the optimum is fairly high. The performance of these decoders is close to the Shannonian limit.

This treatise presents a new and different approach. Instead of serially decoding each of the two trellises in turn, we parallely decode both of them at the same time. As an introduction to this novel technique, let us consider the following example, assuming that we use a simple two-column block interleaver. Figure 3 shows the action of this interleaver with regards to $\mathbf{u}$ and $\mathbf{u}^{(2)}$ for the first seven bits. These seven bits are now encoded with the trellises as depicted in Figure 4. Let us now consider the operation of the decoder. In the first decoding stage, we consider the trellis paths of both trellises that are associated with the dataword bit $u_1$. In the upper trellis, there are only two possible path branches, because the upper trellis commences in the all-zero state. The left-hand-side section of the lower trellis starts also in the all-zero state, and hence there are only two possible paths in this section as well.

We proceed to bit $u_2$. In the upper trellis of Figure 4, there are four possible paths now. In the lower trellis, bit $u_2$ is the input bit to the second encoder belonging to the $(n_1+1)$st trellis stage, since $u_{n_1+1}^{(2)} = u_2$. However, we do not know as yet, which state the second encoder is in after the first $n_1$ transitions, hence the state at the start of the right section in the lower trellis is unkown. We must thus consider two paths emerging from all four possible states, resulting in eight possible paths associated with $u_{n_1+1}^{(2)}$.

Next we consider bit $u_3$. The number of possible paths in the upper trellis of Figure 4 increases to eight. In the lower trellis, bit $u_2^{(2)} = u_3$, i.e. $u_3$ follows $u_1$. There are thus four possible paths in the bottom left section of Figure 4 now.

When we sequentially join the bits $u_4$, $u_5$ and $u_6$, the number of paths in the appropriate sections is doubled each time, corresponding to the logical 0 and 1 values of the bits. In the next decoding stage, i.e. after inputting bit $u_7$, we want to start discarding possible path combinations, which can be excluded from being a part of the optimal codeword. The reason for starting the decoding process only here will become obvious during the following explanation.

The bit combination $(u_1, .., u_7)$ can be considered as being the first seven bits of a dataword that generates a codeword $\mathbf{c}_i$. There are of course many datawords starting with this bit combination and accordingly also many corresponding codewords.

Before we proceed, let us introduce a metric for the parts of the codeword (i.e. for the trellis stages) that are directly associated with the bits $u_1..u_7$. This three-component metric $\mathbf{M}_{\mathbf{c}_i,j\leq 7}$ is the sum of the corresponding forward metric $\mathbf{M}_{\mathbf{x}_i,j\leq 7}$ in the upper trellis in Figure 4, which is also depicted in Figure 5 as quantified by Equation 8, the forward metric $\mathbf{M}_{\mathbf{x}_i^{(2)},m\leq 4}$ for the left section in the lower trellis (see Equation 10 and the intermediate metric $\mathbf{M}_{\mathbf{x}_i^{(2)},n_1\leq m\leq n_1+3}$) for the right section in the lower trellis (see Equation 11):

$$\mathbf{M}_{\mathbf{c}_i,j\leq 7} := \mathbf{M}_{\mathbf{x}_i,j\leq 7} + \mathbf{M}_{\mathbf{x}_i^{(2)},m\leq 4} + \mathbf{M}_{\mathbf{x}_i^{(2)},n_1\leq m\leq n_1+3}. \quad (14)$$

We can now formulate the following algorithm:

**Algorithm**

If several of the trellis paths associated with the input bit combinations $(u_1, .., u_7)$ in the three considered sections of the two trellises exhibit the following properties:

1) their associated paths in the upper trellis terminate in the same state $S_7$ after the seventh transition AND
2) their associated paths in the left section of the lower trellis terminate in the same state $S_4^{(2)}$ after the fourth transition AND
3) their associated paths in the right section of the lower trellis commence in the same state $S_{n_1}^{(2)}$ after the $n_1^{st}$ transition AND
4) their associated paths in the right section of the lower trellis terminate in the same state $S_{n_1+3}^{(2)}$ after the $(n_1+3)^{rd}$ transition,

then only the specific path with the lowest metric $\mathbf{M}_{\mathbf{c}_i,j\leq 7}$ must be kept as a survivor and all others can be discarded in the decoding process. Note that there are four potential path combinations associated with each of the $2^7 = 128$ possible bit combinations, since we do not know the decoder's state in the lower trellis after the $n_1^{st}$ transition.

The reasoning follows exactly the rationale of Section 2. Explicitely, for any complete turbo codeword $\mathbf{c}_i$, its complete metric can be split up as follows:

$$\begin{aligned}\mathbf{M}_{\mathbf{c}_i} &= \mathbf{M}_{\mathbf{x}_i} &&+\mathbf{M}_{\mathbf{x}_i^{(2)}}\\ &= \mathbf{M}_{\mathbf{x}_i,j\leq 7} + \mathbf{M}_{\mathbf{x}_i,7<j} &&+\mathbf{M}_{\mathbf{x}_i^{(2)},m\leq 4}\\ &&&+\mathbf{M}_{\mathbf{x}_i^{(2)},4<m\leq n_1}\\ &&&+\mathbf{M}_{\mathbf{x}_i^{(2)},n_1<m\leq n_1+3}\\ &&&+\mathbf{M}_{\mathbf{x}_i^{(2)},n_1+3<m},\end{aligned} \quad (15)$$

where the various metric components become explicit in Figure 5. The second, fourth, and sixth terms have not been encountered so far; they represent as yet unexplored sections of the trellises namely the right-hand-side section of the upper trellis, the missing intermediate path and the missing right-hand-side section of the lower trellis in Figure 5 respectively, which have all been left blank. Upon rearranging Equation 15, we arrive at the following Equation:

$$\begin{aligned}\mathbf{M}_{\mathbf{c}_i} &= \mathbf{M}_{\mathbf{x}_i,j\leq 7} &&+\mathbf{M}_{\mathbf{x}_i^{(2)},m\leq 4} + \mathbf{M}_{\mathbf{x}_i^{(2)},n_1<m\leq n_1+3}\\ &+ (\mathbf{M}_{\mathbf{x}_i,7<j} &&+\mathbf{M}_{\mathbf{x}_i^{(2)},4<m\leq n_1} + \mathbf{M}_{\mathbf{x}_i^{(2)},n_1+3<m})\\ &= \mathbf{M}_{\mathbf{c}_i,j\leq 7}\\ &+ (\mathbf{M}_{\mathbf{c}_i,7<j}),\end{aligned} \quad (16)$$

where $\mathbf{M}_{\mathbf{c}_i,7<j}$ is the sum of the metrics of the still unexplored sections of the two trellises and hence cannot be evaluated as yet. Explicitely, $\mathbf{M}_{\mathbf{c}_i,7<j}$ is constituted by the backward path metrics of both the upper and lower trellises as well as the metric of the missing central section in Figure 5. Suppose we have two codewords $\mathbf{c}_a$ and $\mathbf{c}_b$, of which the associated paths

a) are different in the three considered sections of the two trellises, which constitute $\mathbf{M}_{\mathbf{c}_i,j\leq 7}$, but exhibit the four criteria 1)..4) listed above, and
b) are identical in all the three unexplored sections of the two trellises, which form part of $\mathbf{M}_{\mathbf{c}_i,7<j}$

The assumption a) implies that $\mathbf{M}_{\mathbf{c}_a,j\leq 7} \neq \mathbf{M}_{\mathbf{c}_b,j\leq 7}$, whereas b) requires that $\mathbf{M}_{\mathbf{c}_a,7<j} = \mathbf{M}_{\mathbf{c}_b,7<j}$ . We have thus $\min\{\mathbf{M}_{\mathbf{c}_a}; \mathbf{M}_{\mathbf{c}_b}\} = \min\{\mathbf{M}_{\mathbf{c}_a,j\leq 7}; \mathbf{M}_{\mathbf{c}_b,j\leq 7}\}$, such that the optimal codeword can never be the one with the higher metric, and this can therefore be discarded. We can repeat this procedure of selecting one of two possible codewords for any pair of codewords exhibiting the properties a) and b). Since the course of the paths does not depend on the bits $u_1..u_7$ outside the three considered sections constituting $\mathbf{M}_{\mathbf{c}_i,j\leq 7}$, we discard from the set of all the codewords sharing properties 1)..4) all those, for which the metric $\mathbf{M}_{\mathbf{c}_i,j\leq 7}$ is not minimal.

When applying the above Algorithm in order to identify the most likely path after the first seven bits, we have to evaluate the metrics of 512 possible paths within the considered sections, since there are $2^7 = 128$ different bit combinations and four possible starting states in the right-hand-side section of the lower trellis. We then have to identify 256 different survivors that differ in at least one of the properties 1)..4), since there are four legitimate states for each property, resulting in $4^4 = 256$ possible survivors. In other words, we can discard the less likely one of two paths sharing the same four properties, reducing the number of possible paths from 512 to 256.

In the following decoding stages, by concatenating a new bit we double the number of possible paths to 512, but since the same four properties still apply, the number of survivors remains 256. Clearly, the above Algorithm constitutes a dynamic programming approach that restricts the number of paths to take into account to 256 at every decoding step.

The trellis states in the four open ends of the two trellises can be amalgamated into a super-state $S_k^*$. Our four properties 1)..4) are therefore uniquely associated with a single super-trellis state $S_7^* = s^*$, and our algortithm has to find the survivor for any possible super-state $s^*$ at every

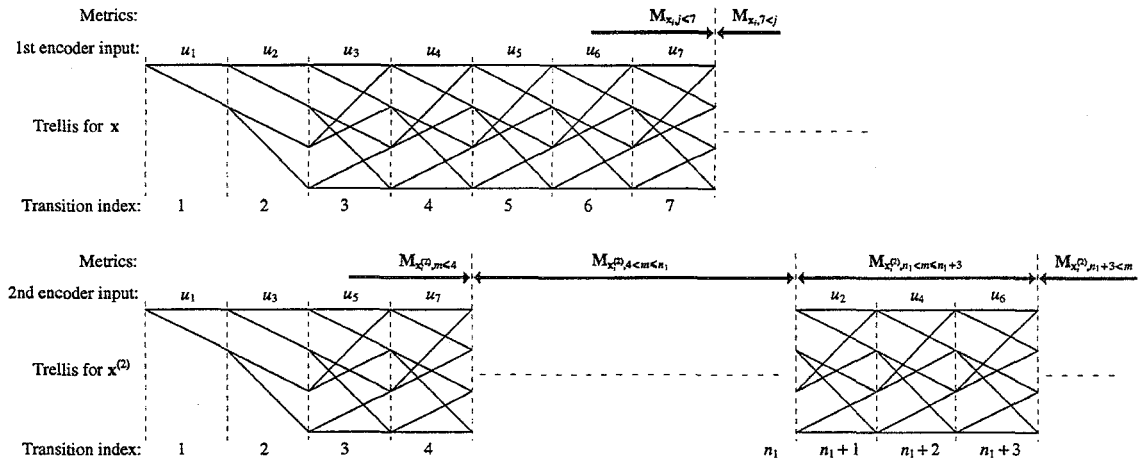


Figure 5: Calculation of the three-component path-metrics for the non-iterative turbo decoder

decoding stage $k$. It can easily be proven that this dynamic programming approach always finds the optimum turbo codeword (in the sense of maximum likelihood).

## 4. ALGORITHMIC PERFORMANCE

We have evaluated the performance of the proposed algorithm in comparison to the best and most complex iterative Turbo decoding algorithm, namely the Maximum Aposteriory (MAP) technique [5]. We carried out simulations using a half-rate, memory-length two RSC code and a 3 columns×333 rows block-interleaver over a Gaussian channel, the results of which are shown in Figure 6. The gap between the iterative MAP Turbo decoder using 16 iterations and the non-iterative 'Flat'-decoder is generally about 0.5dB. In our example using a 4-state convolutional code and a block interleaver of width 2, we have shown that $S_k^*$ can take on 256 different values, i.e. our super-trellis possesses 256 super-states, and 2·256 super-paths have to be treated in each decoding step. Our approach ca be adapted for any interleaver and any convolutional component code, but it is clear, that this complexity becomes prohibitive for more complex Turbo Codes, unless attractive sub-optimum simplifications can be found, which is the subject of our current research.

## 5. CONCLUSION

An optimum non-iterative decoding algorithm for turbo codes was presented and its optimality was shown. As seen in Figure 6, its performance is superior to that of the MAP algorithm, while its complexity is identical to that of a convolutional decoder having the same number of states. Our future work is aimed at reducing the algorithmic complexity of the proposed technique.

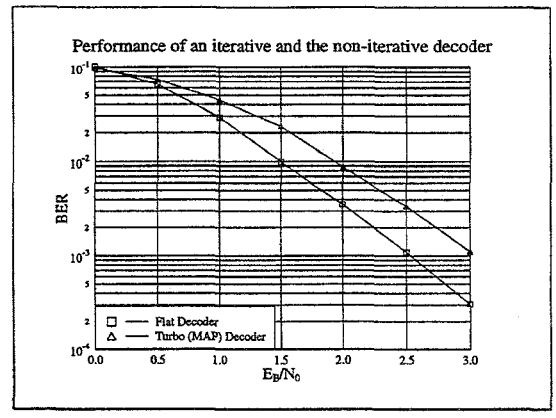


Figure 6: Performance comparison between a conventional Maximum Aposteriory (MAP) decoder using 16 iterations and the proposed optimum turbo decoding scheme (memory-length $M = 2$, $3 \times 333$ block interleaver, each point represents at least 1000 errors)

## 6. REFERENCES


[1] A.J. Viterbi, "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm", in *IEEE Trans. Inform. Theory*, vol. 13, pp. 260..269, 1967.

[2] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-Codes", in *Proc. IEEE Intern. Conference on Communications*, pp. 1064..1070, 1993.

[3] J. Hagenauer and P. Hoeher, "A Viterbi algorithm with soft-decision outputs and its applications," in *IEEE GLOBECOM'89*, (Dallas, Texas), pp. 1680–1686, 1989.

[4] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Transactions on Information Theory*, vol. 42, no. 2, pp. 429–445, 1996.

[5] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain," in *IEEE Intern. Conference on Communications*, pp. 1009–1013, 1995.

[6] P. Jung, "Comparison of turbo-code decoders applied to short frame transmission systems," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 3, pp. 530–537, 1996.