
© 2007 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.
Word Error Probability Estimation by Means of a Modified Viterbi Decoder

Justus Ch. Fricke and Peter A. Hoeher
Information and Coding Theory Lab, Faculty of Engineering
University of Kiel, Kaiserstr. 2, 24143 Kiel, Germany
Phone: +49 431 880 6126, Fax: +49 431 880 6128
E-mail: {jf, ph}@tf.uni-kiel.de

Abstract—In this paper, we introduce a new method for calculating word error probabilities by means of a modified Viterbi decoder. The proposed method can be seen as an approximation of the reliability output Viterbi algorithm (ROVA). The results provided by the proposed algorithm are only slightly worse than that of the ROVA, given a significantly lower computational complexity especially for codes with large memory lengths. As a potential application, both algorithms are compared in reliability-based HARQ.

Index Terms—Modified Viterbi algorithm, word error probability, HARQ

I. INTRODUCTION

In the past, several approaches have been made to provide reliability information on a decoded sequence (rather than on the decoded symbols), i.e., to obtain information how reliable the estimated code word is. In [1] the idea of labeling unreliable paths in the Viterbi decoding process is introduced. If the surviving path is labeled as unreliable, its error probability is considered as too high. This method is often referred to as the Yamamoto-Itoh algorithm (YIA). In [2] the so-called reliability output Viterbi decoder (ROVA) is presented. The ROVA is a modified Viterbi decoder that exactly calculates the word error probability (WEP). In [3] and [4] the reliability information of a symbol-by-symbol soft-output channel decoder is used as an indicator whether the WEP is sufficient or too high. Also based on symbol-by-symbol reliabilities is the approach in [5], where the WEP is approximated from the error probabilities of the info symbols or subblocks contained in the word. All of these reliability estimation techniques have been applied to hybrid automatic repeat request (HARQ) schemes. The estimate of the reliability of the decoded word is used as a criterion whether a packet should be retransmitted or not.

Although appearing similar at the first glance, the first two approaches differ from the latter ones: In [1] and [2] the reliability of a code word—or path through the trellis—is inherently obtained in the decoding process using the Viterbi decoder, while in [3]–[5] the code word reliability is estimated afterwards based on the reliability information provided for the information symbols. The second approach is more flexible, since it can be used with every symbol-by-symbol soft-output decoder. On the other hand, the Viterbi algorithm is known to find the optimum (maximum-likelihood) path through a trellis diagram. Therefore, the Viterbi decoder is more suitable than symbol-by-symbol decoders when trying to obtain word reliabilities.

In this paper, we introduce a new method for estimating word error probabilities by means of a modified Viterbi decoder. The proposed method can be seen as an approximation of the ROVA [2]. In [1] and many methods based on it (e.g., [6] and [7]) metric differences are used to find bounds on the word error probability. In contrast to this, the algorithm in [2] as well as our approach calculates the word error probability as part of the decoding process applying the sequence-estimating property of the Viterbi algorithm.

II. RELIABILITY OUTPUT VITERBI ALGORITHM (ROVA)

Without loss of generality we consider a binary convolutional code with a memory length of $L$ and an information word length of $K$ in the remainder of the paper. The fraction of a trellis in Fig. 1 is used to briefly introduce how the ROVA and the proposed algorithm work. $P(S^n_k)$ denotes the probability that the $n$th state at time index $k$, $S^n_k$, is correct while $P(\overline{S}^n_k)$ is the probability that $S^n_k$ is not correct. $P(S^i_{k-1} \rightarrow S^n_k)$ and $P(S^i_{k-1} \rightarrow \overline{S}^n_k)$ denote the transition probabilities from $S^i_{k-1}$ to $S^n_k$ and $S^i_{k-1}$ to $\overline{S}^n_k$, respectively. The transition probabilities are usually calculated from the received sequence and are assumed to be known in the remainder of the paper.

Figure 1. Trellis fraction for a binary code with transition probabilities and probabilities that the states are correct.

Given the example in Fig. 1, the ROVA calculates $P(S^n_k)$ and $P(\overline{S}^n_k)$ as follows:

$$P(S^n_k) = \frac{P(S^i_{k-1})P(S^i_{k-1} \rightarrow S^n_k)}{\Delta_k} \quad (1)$$
and
\[
\begin{align*}
P(S_k^n) &= \frac{1}{\Delta_k} \left[ P(S_k^{i-1})P(S_k^i \rightarrow S_k^n) 
+ P(S_k^{j-1})P(S_k^j \rightarrow S_k^n) + P(S_k^{\text{even}})P(S_k^i \rightarrow S_k^n) \right] \quad (2)
\end{align*}
\]
The \(\Delta_k\) in the denominator denotes the sum of probabilities over all valid transitions \(\{i, n\}\) connecting a state at time index \(k - 1\) with one at \(k\):
\[
\Delta_k = \sum_{(i,n)} \left[ P(S_k^{i-1}) + P(S_k^{\text{even}}) \right] P(S_k^i \rightarrow S_k^n). \tag{3}
\]
The probabilities \(P(S_k^n)\) and \(P(S_k^{\text{even}})\) can be calculated along with the partial path metrics in the forward recursion. In the final state, \(P(S_{K+1}^n)\) is the probability that the state the surviving path is ending in is not correct. Hence, the error probability of the surviving path is \(P_w = P(S_{K+1}^n)\).

The sum of all possible events (the sum of (1) and (2)) is equal to one. That means that \(P(S_k^n)\) is not a valid path metric for the Viterbi algorithm and two values in addition to the path metric have to be stored for every state.

III. SIMPLIFIED APPROACH

In our approach, subsequently referred to as simplified ROVA, \(P(S_k^n)\) can be calculated along with the add-compare-select operation. Please keep in mind that usually in the Viterbi algorithm metrics rather than probabilities are saved for the selection operation. Please keep in mind that usually in the V iterbi algorithm and two values in addition to the path metric have to be stored for every state.

Instead of state probabilities, we now consider the partial path metrics as displayed in Fig. 2. Let \(\Gamma(S_k^n)\) be the partial path metric of a path ending in state \(S_k^n\).

![Trellis fraction for a binary code with transition probabilities and partial path metrics \(\Gamma(\cdot)\).](image)

Figure 2. Trellis fraction for a binary code with transition probabilities and partial path metrics \(\Gamma(\cdot)\).

Without loss of generality let us assume that the Viterbi decides for the transition from \(S_k^{i-1}\) to \(S_k^n\). The probability that the correct branch is selected becomes
\[
\hat{P}(S_k^n) = \frac{\Gamma(S_k^{i-1})P(S_k^i \rightarrow S_k^n)}{\Gamma(S_k^{j-1})P(S_k^j \rightarrow S_k^n) + \Gamma(S_k^{\text{even}})P(S_k^i \rightarrow S_k^n)}. \tag{4}
\]

In (4) it is assumed that one of the previous states, either \(S_k^{i-1}\) or \(S_k^{j-1}\) is correct, i.e., the probabilities \(P(S_k^{i-1})\) and \(P(S_k^{j-1})\) are ignored. Therefore, (4) is not exact but saves the normalization in (1) and (2). Also, in contrast to (1) and (2), \(P(S_k^n)\) and \(P(S_k^{\text{even}})\) are complementary. As a consequence only one of them has to be calculated and stored.

The same calculation is done for the decision reliabilities of the soft-output Viterbi algorithm (SOVA) in [8]. However, here the interest is only in the probability that the wrong incoming path is selected, not a symbol error probability. Therefore, no updating as in [8] is needed. When tracing back the maximum-likelihood path, the estimated error probability of the decoded word is calculated as
\[
\hat{P}_w = 1 - \prod_{k=1}^{n} \hat{P}(S_k^n), \tag{5}
\]
where \(S_k^n\) denotes the states being part of the surviving path.

The simplified ROVA assumes that one of the incoming paths in a state is the correct one. This would be the case if either the path ending in \(S_{k-1}^i\) or \(S_{k-1}^j\) is correct. Since the Viterbi algorithms finds the maximum-likelihood (ML) path through the trellis, the above assumption is fulfilled if the correct codeword corresponds to the ML path. Therefore, also the calculated error probability, \(P_w\), is exact if the ML path is the correct path. If the ML path contains errors, however, wrong probabilities are multiplied in (5). The longer the error path, i.e., the larger the free distance of the code, the larger the error when the ML path is not the correct path.

Since the original ROVA considers all possible events at every time index \(k\) for the normalization, its calculations do always result in the correct \(P_w\), even if the ML path is not correct.

IV. COMPLEXITY REDUCTION

It can be seen from (4) that the denominator used to normalize the probabilities is much easier to calculate than in case of the ROVA. Also the probability \(P(S_k^n)\) that a state is wrong is not being calculated. A detailed complexity comparison per trellis segment is done in the following. Let \(Q\) be the cardinality of the code symbol alphabet and \(L\) the memory length of the code. Therefore, disregarding the first and the last \(L\) segments of the trellis, one trellis segment consists of \(Q^{L-1}\) states and \(Q\) transitions leaving each state. From (3) \(2Q^{L+1} - 1\) additions and \(2Q^{L+1}\) multiplications arise. For the calculation of (1) and (2) \(2Q^L\) additions and divisions are needed. \(P(S_k^n)\) and \(P(S_k^{\text{even}})\) can be calculated in the forward recursion storing two additional values per state.

The simplified ROVA has two advantages: (4) is normalized in a simpler way and its calculation is based on the partial path metrics of the paths ending in a state instead of its probability. The calculation of the partial path metrics \(\Gamma(S_k^n)\) are calculated by the VA anyway and cause no additional complexity. Therefore, for the calculation of (4) only \(Q - 1\) additions and one division per state are needed. Since there are \(Q^L\) states per trellis segment this results in \(Q^L(2Q - 1)\) additions and
of the simplified ROV A show a small gap of up to word error probabilities, the estimates of the simplified ROV A transmitted via an AWGN channel. Compared to the actual number of states (Q^L) the complexity is reduced compared to the ROVA.

The calculation in (4) is also done for the popular SOVA [8]. That means, the additional complexity caused by the WEP calculation is only one multiplication per trellis segment if a soft-output VA is used. Especially for codes with a large number of states (Q^L) the complexity is reduced compared to the ROVA.

V. NUMERICAL RESULTS

Both decoders, ROVA and simplified ROVA, are optimal in the sense of ML sequence estimation. Therefore, both obtain the same estimate of the transmitted code word. However, while the word error probability P_w calculated by the ROVA is exact, the simplified ROVA calculates only an approximation  \( \hat{P}_w \). In Fig. 3 the two algorithms are compared by means of Monte Carlo simulations. Information words of length \( K = 384 \) are encoded by different convolutional codes and transmitted via an AWGN channel. Compared to the actual word error probabilities, the estimates of the simplified ROVA are too optimistic, i.e., too small. The word error probabilities of the simplified ROVA show a small gap of up to 0.15 dB with respect to the ROVA in the low SNR region. However, that gap reduces for lower word error probabilities. In the context of HARQ, the relevant region is that of a WEP of 0.1 and less. One can also see from Fig. 3 that the gap between the decoders is larger for codes with a larger memory length, which is explained by the longer error paths (see Sec. III).

In the following, a comparison of the two algorithms in a reliability-based ARQ scheme (without packet combining). The polynomials of the FEC code are \((23, 35)_8\). The performance of ROVA and simplified ROVA is evaluated for different values of the target WEP, \( P_{w,t} \). That means, a word is retransmitted if the error probability of the decoded word, \( P_w = P(u \neq \hat{u}) \), is higher than \( P_{w,t} \). Fig. 4 shows the residual WER after ARQ for three different values of \( P_{w,t} \). Before the start of the graphs no successful transmissions (transmission that fulfills \( P_w \leq P_{w,t} \)) were achieved. It can be seen that the error rate in the accepted packets is always below \( P_{w,t} \). It is very close to \( P_{w,t} \) in the low SNR range where high error probabilities are likely. Given the same target error rate \( P_{w,t} \), the resulting residual error rate of the simplified ROVA is slightly higher. The reason for this is the slightly too optimistic estimate \( P_w \) provided by the simplified ROVA.

In Fig. 5 throughput comparisons for the two algorithms are displayed for different values of \( P_{w,t} \). The simplified ROVA achieves a slightly higher throughput. This is a direct consequence of the optimistic estimate of the simplified ROVA: more words are accepted but at a higher error probability. In the higher SNR range, where the simplified ROVA calculates more accurate word error probabilities, the throughput of both algorithms is virtually the same.

The probability mismatch of the simplified ROVA observable from Fig. 3 and Fig. 4 has only a small impact on the performance in terms of residual error rate as well as throughput (see Fig. 4 and Fig. 5). Also, the mismatch could be easily compensated by providing a slightly lower \( P_{w,t} \) to the simplified ROVA. In conclusion, we obtain that the proposed algorithm is only slightly worse than the ROVA, given a significantly lower computational complexity especially for codes with large memory lengths.

<table>
<thead>
<tr>
<th>Table I</th>
<th>COMPARISON OF ADDITIONAL COMPLEXITY PER TRELLIS SEGMENT (COMPARSED TO VA) OF ROVA AND SIMPLIFIED ROVA.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROVA</td>
<td>simpl. ROVA</td>
</tr>
<tr>
<td>Additions</td>
<td>( 2Q^L(Q + 1) - 1 )</td>
</tr>
<tr>
<td>Multiplications</td>
<td>( 2Q^{L+1} )</td>
</tr>
<tr>
<td>Divisions</td>
<td>( 2Q^L )</td>
</tr>
</tbody>
</table>

Figure 3. Word error probability estimates for different convolutional codes. The results are the average word error probabilities provided by the two decoders under investigation.

Figure 4. Residual WER after decoding for different target word error probabilities \( P_{w,t} \). The code polynomials are \((23, 35)_8\).
VI. Conclusions

In this paper, a modified Viterbi algorithm was presented. The algorithm can be considered as an approximation of the ROVA [2], which calculates the error probability of a decoded sequence. The accurateness of the word error probability calculation is very close to that of the original ROVA. However, the complexity of the presented algorithm is much lower than that of the ROVA, especially for large memory lengths. If combined with the soft-output Viterbi algorithm [8], the extra complexity caused by the WEP calculation of the proposed algorithm is marginal.

It should be also noted that decoding is not the only application. The proposed algorithm can be used for any type of sequence estimation the Viterbi algorithm is applicable to (e.g. equalization and detection).

ACKNOWLEDGMENT

The authors would like to thank Dr. Ingmar Land for valuable discussions and remarks on Viterbi decoding.

REFERENCES