Low-Complexity ML Decoding for Convolutional Tail-Biting Codes

Hung-Ta Pai, *Member, IEEE*, Yunghsiang S. Han, *Senior Member, IEEE*, Ting-Yi Wu, Po-Ning Chen, *Senior Member, IEEE*, and Shin-Lin Shieh

***Abstract*—Recently, a maximum-likelihood (ML) decoding algorithm with two phases has been proposed for convolutional tail-biting codes [1]. The first phase applies the Viterbi algorithm to obtain the trellis information, and then the second phase employs the algorithm A* to find the ML solution. In this work, we improve the complexity of the algorithm A* by using a new evaluation function. Simulations showed that the improved A* algorithm has over 5 times less average decoding complexity in the second phase when $E_b/N_0 \geq 4$ dB.**

***Index Terms*—Viterbi algorithm, maximum-likelihood, tail-biting codes, algorithm A*.**

I. Introduction

CONVOLUTIONAL Tail-Biting Codes (CTBC) can overcome the loss on the code rate, and induces less performance degradation [2], [3]. In the trellis of CTBC, there is a one-to-one correspondence between a codeword and a path with the same initial and final state, which is called a *tail-biting path*.[1] If the number of initial states (equivalently, final states) of the convolutional tail-biting code is N_s, the trellis is composed of N_s subtrellises with the same initial and final state. These subtrellises are called *tail-biting subtrellises*, or simply subtrellises, and will be denoted by T_i for the ith subtrellis.

Several suboptimal decoding algorithms for the CTBC have been proposed [2], [4]. Among them, the wrap-around Viterbi algorithm (WAVA) is the one with the least decoding complexity [2].

A straightforward optimal decoding algorithm for the CTBC codes is to perform the Viterbi algorithm on all of the tail-biting subtrellises; however, such approach may be impractical due to its high computational complexity. Recently, an ML decoding algorithm of practical decoding complexity has been proposed [1]. This scheme has two phases. In the first phase, the Viterbi algorithm (VA) is applied to the trellis of the convolutional tail-biting code to obtain the trellis information. Based upon the trellis information, the algorithm A* is then performed on all subtrellises in the second phase to yield the ML decision. It has been shown that the decoding complexity can be reduced from N_s VA trials to equivalently 1.3 VA trials without sacrificing the optimality in performance.

In this work, an improved algorithm A* with a new evaluation function is presented. Simulations showed that the complexity in the second phase can be further reduced down to 1/5 of the original scheme at medium-to-high signal to noise ratio (SNR).

II. Maximum-Likelihood Decoding of the CTBC Using Improved Algorithm A*

Let $\mathcal{C}$ be an $(n, 1, m)$ convolutional tail-biting code of L information bits, where n is the number of output bits per information bit, and m is the memory order. Hence, the trellis of $\mathcal{C}$ has $N_s = 2^m$ states at each level, and is of $L + 1$ levels. As aforementioned, the corresponding tail-biting paths for codewords of $\mathcal{C}$ should constrain on the same initial and final state. By relaxing such constraint, we denote the super code of $\mathcal{C}$, which consists of all paths that may end at a final state different from the initial state, by $\mathcal{C}_s$.

Denote by $\boldsymbol{v} \triangleq (v_0, v_1, \ldots, v_{N-1}) \in \{0,1\}^N$ the binary codeword of $\mathcal{C}$, where $N = nL$. Define the hard-decision sequence $\boldsymbol{y} = (y_0, y_1, \ldots, y_{N-1})$ corresponding to the received vector $\boldsymbol{r} = (r_0, r_1, \ldots, r_{N-1})$ as

$$y_j \triangleq \begin{cases} 1, & \text{if } \phi_j < 0; \\ 0, & \text{otherwise,} \end{cases}$$

where $\phi_j \triangleq \ln[\Pr(r_j|0)/\Pr(r_j|1)]$. Then, it can be derived by the Wagner rule that the ML decoding output $\boldsymbol{v}^*$ for received vector $\boldsymbol{r}$ satisfies

$$\sum_{j=0}^{N-1} (v_j^* \oplus y_j)|\phi_j| \leq \sum_{j=0}^{N-1} (v_j \oplus y_j)|\phi_j| \quad \text{for all } \boldsymbol{v} \in \mathcal{C},$$

where "$\oplus$" is the exclusive-or operation. We thereby define a new metric for the path in a subtrellis as follows.

Definition 1: For a path with zero-one labels $\boldsymbol{x}_{(\ell n-1)}^{(i)} = (x_0^{(i)}, x_1^{(i)}, \ldots, x_{\ell n-1}^{(i)})$, which ends at level ℓ in subtrellis T_i, define the *metric* associated with it as

$$M\left(\boldsymbol{x}_{(\ell n-1)}^{(i)}\right) \triangleq \sum_{j=0}^{\ell n-1} M(x_j^{(i)}),$$

where $M(x_j^{(i)}) \triangleq (y_j \oplus x_j^{(i)})|\phi_j|$ is the *bit metric*.

The metrics for those paths not belonging to any subtrellis T_i, where $1 \leq i \leq N_s$, can be similarly defined.

In the first phase, the VA is applied using the metric just defined. Then, we will have a set of N_s survivors ending at the final states after phase one. Notably, these survivor paths correspond to codewords in $\mathcal{C}_s$, but not necessarily codewords

Manuscript received December 27, 2007. The associate editor coordinating the review of this letter and approving it for publication was R. Nabar. This work was supported by the National Science Council, Taiwan, R.O.C., under the project No. NSC 96-2628-E-305-001 and NSC 96-2221-E-305-003.

H.-T. Pai, Y. S. Han, and T.-Y. Wu are with the Graduate Institute of Communication Engineering, National Taipei University (e-mail: {htpai, yshan}@mail.tnpu.edu.tw, mavericktywu@gmail.com).

P.-N. Chen is with the Dept. of Communications Engineering, National Chiao Tung University (e-mail: qponing@mail.nctu.edu.tw).

S.-L. Shieh is with Sunplus mMobile Inc., Hsinchu 300, Taiwan, R.O.C., and also the Dept. of Communications Engineering, National Chiao Tung University (e-mail: shinlinshieh@yahoo.com.tw).

Digital Object Identifier 10.1109/LCOMM.2008.072181.

[1] Hence, "tail-biting paths" and "codewords" are interchangeably used in this work.

in $\mathscr{C}$. We also retain the metric of the survivor ending at state s_ℓ of level ℓ, obtained in phase one, and will denote it by $c(s_\ell)$.

Instead of operating on the entire trellis with respect to the super code $\mathscr{C}_s$, the decoding of the algorithm A* only operates on tail-biting subtrellises in the second phase. Thus, the output of the second phase will always be a codeword in $\mathscr{C}$. For each path with zero-one labels $\boldsymbol{x}^{(i)}_{(\ell n-1)}$ over subtrellis T_i, a new evaluation function f is associated with it as follows:

$$f(\boldsymbol{x}^{(i)}_{(\ell n-1)}) = g(\boldsymbol{x}^{(i)}_{(\ell n-1)}) + h(\boldsymbol{x}^{(i)}_{(\ell n-1)}),$$

where

$$g(\boldsymbol{x}^{(i)}_{(\ell n-1)}) = g(\boldsymbol{x}^{(i)}_{((\ell-1)n-1)}) + \sum_{j=(\ell-1)n}^{\ell n-1} M(x^{(i)}_j) \quad (1)$$

with initial value $g(\boldsymbol{x}^{(i)}_{(-1)}) = 0$,

$$h(\boldsymbol{x}^{(i)}_{(\ell n-1)}) = \max\{0, c(s_L) - c(s_\ell)\}$$

and s_ℓ and s_L are the states that paths $\boldsymbol{x}^{(i)}_{(\ell n-1)}$ and $\boldsymbol{x}^{(i)}_{(N-1)}$ respectively end at. It is easy to see that $f(\boldsymbol{x}^{(i)}_{N-1}) = g(\boldsymbol{x}^{(i)}_{N-1})$ since $h(\boldsymbol{x}^{(i)}_{N-1}) = \max\{0, c(s_L) - c(s_L)\} = 0$; hence, the tail-biting path with the minimum f-function value is exactly the one with the minimum ML metric. Moreover, since $c(s_\ell)$ is the minimum metric among all paths that start from any initial state but end specifically at state s_ℓ of level ℓ, we have

$$c(s_\ell) \le c(s_{\ell-1}) + \sum_{j=(\ell-1)n}^{\ell n-1} M(x^{(i)}_j),$$

where $s_{\ell-1}$ and s_ℓ are respectively the states that paths $\boldsymbol{x}^{(i)}_{((\ell-1)n-1)}$ and $\boldsymbol{x}^{(i)}_{(\ell n-1)}$ end at. Hence, f is non-decreasing along any tail-biting path in subtrellis T_i.

Then, equipped with an *Open Stack* for paths visited thus far by the Algorithm A*, and a *Close Table* for starting and ending states and ending level of the paths that have ever been on top of the Open Stack, we summarize the Improved Algorithm A* on subtrellises in the following.

Step 1. *Sort all survivors found in phase one according to ascending order of their metrics. If the survivor with the least metric is also a tail-biting path (that starts and ends at the same state), then output it as the final ML decision, and the algorithm stops.*

Step 2. *Set ρ equal to the least metric among all survivors that are also tail-biting paths, if such exists; otherwise, set $\rho = \infty$.*

Step 3. *Delete all survivors whose metrics are equal to or greater than ρ.*

Step 4. *Load into the Open Stack all zero-length pathes that start at the same states as the ending states of the remaining survivors. Sort these zero-length paths in the Open Stack according to ascending order of their f-function values.*

Step 5. *If the Open Stack is empty, output the survivor with metric ρ as the final ML decision, and the algorithm stops.*

Step 6. *If the top path in the Open Stack reaches level L in its subtrellis, output the path as the final ML decision, and the algorithm stops.*

Step 7. *If the information of the starting and ending states and ending level of the top path has been recorded in the Close Table, discard the top path from the Open Stack, and go to Step 5; otherwise, record the paired information of the starting and ending states and ending level of the top path in the Close Table.*

Step 8. *Compute the f-function values of the successors of the top path, and delete the top path from the Open Stack. If the f-function value of any successor is equal to or greater than ρ, just delete it.*

Step 9. *Insert the remaining successor paths into the Open Stack, and re-order the Open Stack according to ascending f-function values. Go to Step 5.*

The optimality of the above algorithm can be substantiated by the fact that Step 7 will never delete the true ML decision.

Suppose that at Step 7, the paired information of the starting and ending states and ending level of the new top path $\boldsymbol{x}^{(i)}_{(\ell n-1)}$ has been recorded in the Close Table at some previous time due to path $\hat{\boldsymbol{x}}^{(i)}_{(\ell n-1)}$. Since path $\boldsymbol{x}^{(i)}_{(\ell n-1)}$ must be the offspring of a path $\boldsymbol{x}^{(i)}_{(\bar{\ell} n-1)}$ that once coexisted with $\hat{\boldsymbol{x}}^{(i)}_{(\ell n-1)}$ in the Open Stack at the time $\hat{\boldsymbol{x}}^{(i)}_{(\ell n-1)}$ was on top of the Open Stack, where $\bar{\ell} < \ell$, we have

$$f\left(\boldsymbol{x}^{(i)}_{(\ell n-1)}\right) \ge f\left(\boldsymbol{x}^{(i)}_{(\bar{\ell} n-1)}\right) \ge f\left(\hat{\boldsymbol{x}}^{(i)}_{(\ell n-1)}\right).$$

Notably, the first inequality follows from the non-decreasingness of the f-function values along any path in subtrellis T_i, and the second inequality is valid because the top path in the Open Stack always carries the minimum f-function value. As a result, the minimum-metric tail-biting path generated from path $\boldsymbol{x}^{(i)}_{(\ell n-1)}$ will always have an equal or larger metric than the minimum-metric tail-biting path generated from path $\hat{\boldsymbol{x}}^{(i)}_{(\ell n-1)}$. The deletion of path $\boldsymbol{x}^{(i)}_{(\ell n-1)}$ accordingly will not eliminate the ML tail-biting path.

Similar argument can be used to prove that the first top path that reaches level L shall have the minimum metric among all tail-biting paths generated from the remaining paths coexisted with this top path. The optimality of the algorithm is therefore confirmed.

III. Simulation Results over AWGN Channel

In this section, we investigate the computational effort and the word error rate (WER) of the proposed decoding algorithm by simulations over the additive white Gaussian noise (AWGN) channel with BPSK-modulated inputs. The $(2, 1, 6)$ binary convolutional tail-biting code with generator $155, 177$ (octal) is considered. The length of the information bits used in our simulations is 48. We will respectively abbreviate the proposed algorithm and the algorithm given in [1] as IA* and A* in the sequel. For all simulations, at least 30 word errors have been reported to ensure that there is no bias on the simulation results.

In Figure 1, we compare the WERs of the IA* with those obtained by the A*, as well as the WAVA given in [2] with two iterations. Since both the IA* and the A* are ML

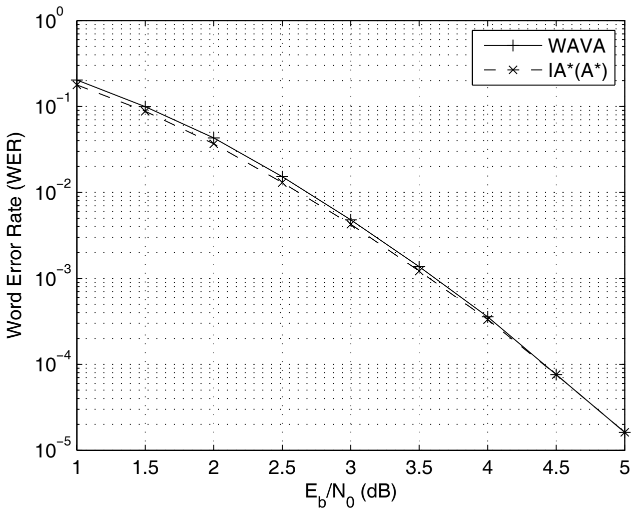

Fig. 1. The word error rates (WERs) of IA*, A*, and WAVA.

TABLE I
COMPARISON OF AVERAGE (AVE) AND MAXIMUM (MAX) NUMBERS OF BRANCH METRIC COMPUTATIONS IN PHASE TWO

E_b/N_0	3 dB		4 dB		5 dB	
algorithm	ave	max	ave	max	ave	max
WAVA	3072	3072	3072	3072	3072	3072
A*	335	29313	221	11930	165	11010
IA*	94	11045	42	3427	27	780

decoders, it is reasonable that they yield the same WER. Also noted from Figure 1 is that the WAVA provides near-optimal WER performance, and is only slightly inferior to the optimal performance when E_b/N_0 is between 1 dB and 4.5 dB.[2]

In Table I, we compare the average and maximum computational efforts of the IA* with those of the A* and the WAVA in phase two. For a fair comparison, only the computations of branch metrics, i.e., the second term in (1), are considered.[3] Since the WAVA searches the entire trellis in phase two, its number of branch metrics computed is the same for all SNRs. By the simulation results, the IA* has a much smaller average and maximum computational complexity than the A* for all SNRs. For example, the average computational effort of the IA* is about 7 times and 70 times less than that of the A* and the WAVA, respectively, when $E_b/N_0 \geq 4$ dB as shown in Table I.

REFERENCES

[1] P. Shankar, P. N. A. Kumar, K. Sasidharan, B. S. Rajan, and A. S. Madhu, “Efficient convergent maximum likelihood decoding on tail-biting,” available at http://arxiv.org/abs/cs.IT/0601023.

[2] R. Y. Shao, S. Lin, and M. P. C. Fossorier, “Two decoding algorithm for tailbiting codes,” *IEEE Trans. Commun.*, vol. COM-51, no. 10, pp. 1658–1665, Oct. 2003.

[3] Q. Wang and V. K. Bhargava, “An efficient maximum likelihood decoding algorithm for generalized tail biting,” *IEEE Trans. Commun.*, vol. COM-37, no. 8, pp. 875–879, 1989.

[4] R. V. Cox and C. E. W. Sundberg, “An efficient adaptive circular viterbi algorithm for decoding generalized tailbiting convolutional codes,” *IEEE Trans. Veh. Technol.*, vol. 43, no. 11, pp. 57–68, Feb. 1994.

[2] E_b/N_0 denotes the signal-to-noise ratio per information bit.

[3] The proposed recursive implementation of the Algorithm A* in [1] has a merit that no branch metric computation is required when the successor path has the same f-function value as its predecessor. By this reason, the complexities of the IA* in Table I also excludes the computations of those branch metrics that equate the f-function values of the successor and its immediate predecessor.