A New Bidirectional Algorithm for Decoding Trellis Codes

Vojin Šenk[1] and Predrag Radivojac[2]
1) University of Novi Sad, School of Engineering, Yugoslavia
2) Temple University, Center for Information Sciences and Technology, U. S. A.
Email: ram_senk@uns.ns.ac.yu, predrag@snowhite.cis.temple.edu

Abstract

A new parallel procedure for decoding trellis codes with large constraint length is simulated and compared to the bidirectional stack algorithm. The systolic organization of processing units arranged in two mutually connected arrays enables decoding effort unaffected by single correctable bursts of errors whose branch length is not greater than some $\mathcal{T}$, where $\mathcal{T}$ is not greater than the code memory length. During the course of decoding, the algorithm produces a set of tentative decisions of increasing reliability, until it reaches the final decision. Every tentative decision is composed of a portion of a forward and backward path, connected via a tunnel of length $\mathcal{T}$. After each new tentative decision is made, a new set of discarding criteria is produced. According to these criteria, a vast number of partially explored paths is discarded from all the stacks, speeding up the decoding procedure. The results show significant reduction in decoding effort (measured by the number of extended paths in parallel) compared to known sequential procedures.

1. Introduction

Decoding trellis codes is a complex process, since there is no algebraic decoding algorithm (similar to those existent for block codes) whose application does not require a restriction to codes of very poor quality. Two of the most powerful techniques for decoding trellis codes are the suboptimum tree search of sequential decoding, such as the stack (or ZJ) algorithm [1], and the optimal trellis search of Viterbi decoding [1]. Both the above-mentioned techniques attempt to find the best path $\hat{\mathbf{I}} = [\hat{\mathbf{i}}_1, \hat{\mathbf{i}}_2, \ldots, \hat{\mathbf{i}}, \ldots, \hat{\mathbf{i}}_L]$, $\dim(\hat{\mathbf{i}}_n) = K$, through a graph (tree or trellis) in which the branches are assigned "branch metrics" μ_n. The branch metric is essentially the scaled log-likelihood function between the coded symbols $\mathbf{x}_n$, $\dim(\mathbf{x}_n) = N$, of the branch corresponding to a possible data input $\hat{\mathbf{i}}_n$, and the channel output $\mathbf{y}_n$, $\dim(\mathbf{y}_n) = N$, corresponding to the actual data input symbol $\mathbf{i}_n$, $\dim(\mathbf{i}_n) = K$. For sequential decoding, this metric is called the Fano metric. The branch metrics are cumulative so that, over a graph of length $L +$ branches (the transmission is framed, i.e. every information block consisting of L K-tuples is terminated by a tail of another , being the code memory), the objective of both techniques is to find the path $\hat{\mathbf{I}}$ for which the total metric is maximum over all possible transmitted paths. For all input data equally likely, the decoded sequence error probability is minimized. In the rest of the paper we assume $K = 1$.

2. The New Algorithm

In order to facilitate the discussion in the sequel, we will restrict ourselves to the Binary Symmetric Channel (BSC) and alternatively use the notions of the Fano metric and the accumulated Hamming distance of the (partly explored) path through the tree. These two parameters are, after suitable scaling, tied via

$$\mu_{(\text{path})} = l_{(\text{path})} - A \cdot d_{(\text{path})}, \quad A \in \mathbb{R}^+, \tag{1}$$

where l is the length of the path, and d is the accumulated distance between the path and the corresponding portion of the channel output sequence. $\mathbb{R}^+$ is the set of positive real numbers. Like the bidirectional stack algorithm [2] [3] [4] [5], the new algorithm is also based on the following notions

- the reverse trellis code;
- the tunnel;
- the tentative decision;
- a set of discarding criteria.

The reverse trellis code is obtained from the original one by time reversing. When plotted from right to left, the corresponding trellis diagram of the reverse code is the same as the original one, with the arrows on all the branches pointing in the backward direction. The tunnel is the unique sequence $\mathcal{T}$ ($0 \le \mathcal{T} \le$) branches long that connects two states in the trellis. The tentative decision is the information sequence $L +$ branches long that connects the known *initial* and *terminal* trellis states (direction does not matter here) that has the highest accumulated metric of all the sequences of that length analyzed so far. A set of discarding criteria is a means to tell beforehand whether a partly explored path is likely to be a part of the finally decoded sequence or not (in the latter case, the path may be eliminated from the subsequent search).

The algorithm uses $2 \cdot (L + \quad - \mathcal{T})$ processing units, although in practice, this number can be somewhat smaller. Half of the processing units are intended for forward search while the others, using the reverse code, perform backward search. Processing units, shown in Fig. 1, are arranged in two arrays so that the processing unit at depth l deals only with paths of depths $l - 1$, l, and $l + 1$ (except for the processors at the ends of each array). Each processing unit contains a stack of size M_l (l being the depth of the unit, $l = 1, 2, \ldots, L + \quad - \mathcal{T}$).

The steps of the algorithm are:

1. Put the root node into the first stack on the F side, and the (unique) terminal node into the first stack on the opposite side, associating them the zero metric.

0-7803-6490-2/01/$10.00 ©2001 IEEE

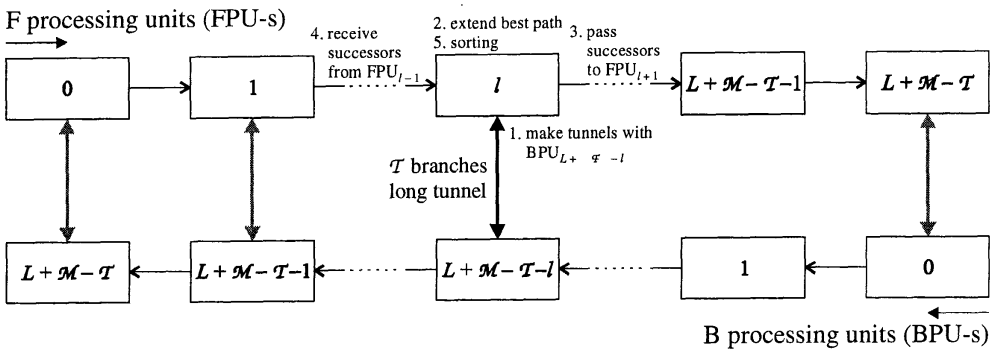

Figure 1. *Organization of the new bidirectional decoder*

The steps 2 – 5 are performed simultaneously in all processing units from both directions. We describe the behavior of the unit at depth l.

2. Choose the node with the best metric and eliminate it from the stack. Link it via a tunnel (if a tunnel is possible, i.e. if the states match) to each of existing paths in the stack at depth $L-l+\quad-\mathcal{T}$. (If a tunnel is branches long, then the best path from the stack can be linked to all the paths from the stack at depth $L-l$). The total length of the paths obtained in this way is $l+\mathcal{T}+(L-l+\quad-\mathcal{T})=L+\quad$ branches. Store the best combined path into the tentative decision register. If there is already a path in the register, keep the better. Prune the paths remaining in all stacks according to any of discarding criteria established. If all the stacks are emptied in this way, output the tentative decision as the decoder's final decision and terminate the algorithm.
3. Calculate the metrics of all the successors of the processed path, and eliminate all of them that do not conform to the discarding criteria established.
4. a) Pass remaining successors to the processing unit at depth $l+1$.
   b) Receive all the successors from depth $l-1$ and sort them according to their metrics.
5. Return to step 2.

Note that steps 4a and 4b are performed in reverse order by neighboring processing units. In such a case processing units at even depths perform step 4a simultaneously with processing units at odd depths performing step 4b. In the second part of step 4 even and odd-depth processing units switch their roles.

After each new tentative decision is formed, several discarding criteria for the paths stored in all processing units can be established. The first one is based on the nonselection principle [1], and it states that from the two paths diverging from the same node, the ZJ algorithm keeps the one whose minimum Fano metric until the end node is maximal, and not the one with the maximum Fano metric at the end node. Since this does not increase the probability of error significantly [1], all the paths from any direction that do not satisfy this criterion when compared with the tentative decision may be discarded.

Another type of discarding is based on the path distance. Denoting by $d^{\text{inv}}(L-l+\quad-\mathcal{T})$ the minimum path metric from any stack from the reverse direction at depths from $L-l+\quad-\mathcal{T}$ to $L+\quad-\mathcal{T}$, a path of length l and distance d can be discarded whenever $d+d^{\text{inv}}(L-l+\quad-\mathcal{T})\geq d_{\text{TD}}$, where d_{TD} is the total accumulated distance of the tentative decision. This is a maximum likelihood criterion, and used alone produces the MLD result. Moreover, a path may be discarded whenever it is ranked below the M_l-th place in its current stack (as in the M-algorithm [6]), or when the metric difference between the best path ever from a stack and the path in question is greater than T_l (as in the T-algorithm [7]). The last two discarding criteria provide small number of paths in all stacks.

The algorithm may also be terminated after the assigned time for its execution has elapsed. Since the algorithm in both directions easily follows the correct path till it meets an error burst, and since the distribution of errors in the tunnel does not affect the performance in step 2, no correctable error pattern confined to $\mathcal{T}$ successive branches can affect the number of steps to correct tentative decision (the one obtained by the VA). This would stimulate the choice of longer tunnels if there were not the negative effect of pursuing a great number of unnecessary tunnels when $\mathcal{T}$ approaches .

Although the space complexity of the new algorithm is linear with the information frame length, there are some practical difficulties with passing on the discarding criteria to all the processing units and analyzing new candidates for the path stored in the tentative decision register. These problems may be dealt easily if the transfer of these data is pipelined. A certain delay in passing on discarding criteria and candidate tentative decisions thus imposed would mostly influence the moment termination and yield results that are slightly worse than shown in this paper.

3. Simulation results

In order to assess the performance of the algorithm we have first constructed bidirectional optimum distance profile codes (i.e. codes with maximum bidirectional column distance (BCD) growth, where BCD at each depth equals minimum of forward and backward column

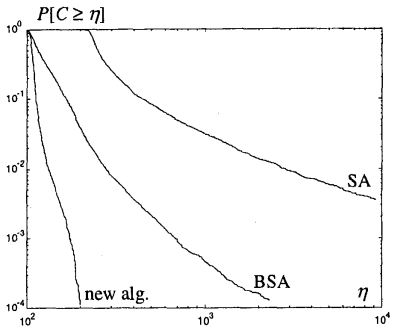

Figure 2. *Computational distributions for stack algorithm, bidirectional stack algorithm ($\mathcal{T}$= 20), and new algorithm at the moment of reaching the final decision*

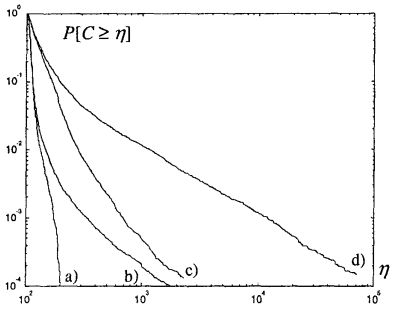

Figure 3. *Computational distribution a) new alg. – moment of final decision; b) new alg. – moment of termination; c) BSA – moment of final decision; d) BSA – moment of termination*

distances), and then conducted a simulation, assuming Binary Symmetric Channel, = 32 bidirectional code with $K = 1$, $N = 2$. Generator polynomials in their usual octal form are $\mathbf{g}_1 = 45107770451$ and $\mathbf{g}_2 = 60235556203$; and the column distance function is BCDF = [2, 3, 3, 4, 4, 5, 5, 6, 6, 6, 7, 7, 8, 8, 8, 8, 9, 9, 9, 10, 10, 10, 10, 11, 11, 11, 11, 12, 12, 12, 13, 13]. The size of the information block is $L = 180$. The crossover probability for BSC is set to $p = 0.045$, or $E_b/ _0 = 4.6$ dB, meaning that the code rate equals the cutoff rate of the channel.

Figure 2 shows the computational distributions obtained for the unidirectional stack algorithm, bidirectional stack algorithm, and the new algorithm (the latter two used $\mathcal{T}$= 20). In all cases we allowed unlimited stack size, i.e. we took $M_l \to \infty$, as well as $T_l \to \infty$, $\forall l$. C represents the number of parallel steps of the algorithm. In each step every processing unit performs two metric computations.

In Fig. 3 we show the computational distributions of the BSA and the new algorithm at the moments of reaching the last decision and regular termination of the algorithm. Reaching the last decision describes the moment in which the final decision is already in the tentative decision register, meaning that terminating the algorithm in any moment after this could not induce additional errors. This can yield another terminating condition, imposing a limit on the number of computations. Namely it is readily observed that the number of computations needed to reach the (correct) final decision is substantially lower than the number met waiting for all non-discarded paths to die out.

Note that the algorithm by Kallel and Lee [5], that is much faster than the original stack algorithm, implies a tunnel of zero length, and that it is already slower than the variation used here with $\mathcal{T}$= 20, implying that correctable error bursts of length not larger than $\mathcal{T}$ are passed in one (tunneling) step. These bursts are the dominant cause of the delay of the original stack algorithm, as well as the one proposed by Kallel and Lee.

Our new algorithm is considerably faster than BSA in its form introduced in [2] and [3] using tunnels, and is still faster if the moment of storing the final decision into the tentative decision register is reached. This moment is easily checked introducing a simple CRC check of the information sequence before applying the convolutional code, and checking the tentative decision whether the CRC condition is met. Otherwise, the algorithm can be terminated using a computational limit that could be evaluated experimentally.

4. Conclusion

In this paper a new parallel bidirectional algorithm for decoding trellis codes is proposed, simulated, and compared to unidirectional and bidirectional stack algorithm. Analyzing the results obtained, we observe that the new algorithm is not only faster than the bidirectional stack algorithm by Kallel and Li [5], but also than the tunneled version introduced in [2] and [3].

References

[1] J. L. Massey, "Error Bounds for Tree Codes, Trellis Codes and Convolutional Codes with Encoding and Decoding Procedures", CISM Courses and Lectures No. 216, Coding and Complexity, 1975.

[2] V. Šenk, P. Radivojac, "The Bidirectional Stack Algorithm," *Proc. 1997 IEEE Int. Symp. Inf. Th.*, ISIT'97, p. 500, Ulm, Germany, July 1997.

[3] V. Šenk, P. Radivojac, "The Bidirectional Stack Algorithm - Simulation Results," *Proc. 2-nd Conf. Telecomm. in Modern Satellite and Cable Services*, pp. 349-352, Niš, Yugoslavia, October 1995.

[4] V. Šenk, The error exponent of particular families of channel codes with suboptimum decoding procedures that enable its attainment, Ph. D. thesis, Univ. of Belgrade, Dept. of Engin., 1992. (in Serbian)

[5] S. Kallel and K. Li, "Bidirectional Sequential Decoding," *IEEE Trans. Inf. Th.*, vol. IT-43, No. 4, pp. 1319-1326, July 1997.

[6] F. Jelinek and J. B. Anderson, "Instrumentable Tree Encoding of Information Sources," *IEEE Trans. Inf. Th.*, vol. IT-22, pp. 82-83, January 1971.

[7] S. J. Simmons, "Breadth First Trellis Decoding with Adaptive Effort," *IEEE Trans. Comm.*, vol. 38, pp. 3-12, January 1990.