An Application of the Bidirectional Stack Algorithm to Speech Coding

Vojin Šenk, Predrag Radivojac, and Ognjen Todić

Abstract— **Performance of the bidirectional metric-first tree encoding algorithm is measured on real speech data coded at 1 bps. The main drawback of the unidirectional stack algorithm, variability of its decoding effort as well as decoding erasures, is slightly alleviated. Comparisons, with respect to SNR and time complexity, with unidirectional stack algorithm (SA) and the M-algorithm (MA) are made. A procedure for nonlinear bidirectional code design, based on SGL algorithm is proposed.**

Keywords— **Tree encoding, sequential algorithms, bidirectional tree search, nonlinear bidirectional trellis code design**

I. Introduction

Trellis coding is a proven technique for source coding. It can be considered in terms of an encoder-decoder pair. Decoder consists of a finite state machine driving a table-lookup codebook of reproduction values. Encoder is a trellis search algorithm that chooses the channel sequence so as to minimize the distortion between the input sequence and the decoder output sequence. Whether fixed or time-varying, trellis codes can be most conveniently described and analyzed by means of the trellis diagram. Figure 1 shows a trellis source decoder and Fig. 2 shows the corresponding trellis diagram for the binary trellis code with $\mathcal{M}$ delay elements and a delayless transformation.

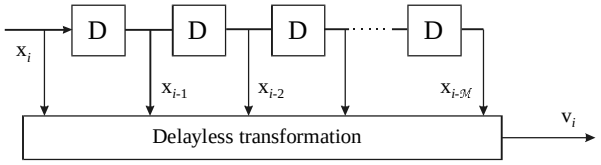

Fig. 1. Trellis decoder

We assume a q-ary trellis code with n source and destination symbols per branch, resulting in a code rate

$$R = \frac{1}{n}\log(q) \quad \text{bits/source symbol}, \tag{1}$$

Vojin Šenk is with the University of Novi Sad, Faculty of Engineering, Trg Dositeja Obradovića 6, 21000 Novi Sad, Yugoslavia, ram_senk@uns.ns.ac.yu

Predrag Radivojac is with the University of Novi Sad, Faculty of Engineering, Trg Dositeja Obradovića 6, 21000 Novi Sad, Yugoslavia, radivojac@uns.ns.ac.yu

Ognjen Todić is with the Stanford University, Department of Electrical Engineering,161 Packard, 350 Serra Mall, Stanford, CA 94305-9505, todic1@stanford.edu

where log (·) stands for $\log_2(\cdot)$. This means that for each q-ary input, from the channel alphabet $\mathcal{X} = \{c_1, c_2, ..., c_q\}$, the trellis source decoder emits n symbols from the user alphabet $\mathcal{V} = \{b_1, b_2, ..., b_B\}$, and the sequence of input symbols defines a path-map in the trellis diagram. The trellis is assumed to be initiated and terminated in **0** state (there are $q^{\mathcal{M}\log(q)}$ states), and we let the total code length be L branches followed by the $\mathcal{M}$ branches in the tail. We also assume the same source and destination alphabet, i.e. $\mathcal{U} = \{a_1, a_2, ..., a_A\} = \mathcal{V}$.

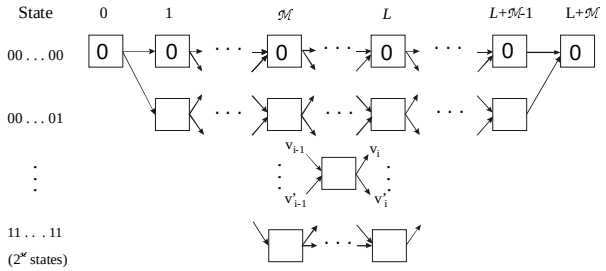

Fig. 2. Trellis diagram, $q = 2$

The source encoder searches for that path in the trellis whose destination (i.e. user) sequence **v** most closely resembles the source sequence **u**. Once a source encoder picks a path, it sends q-ary symbols x through a channel. Channel is assumed to be noiseless and its output sequence **x** drives the trellis source decoder through the sequence of states producing the trellis source decoder output. This procedure is equivalent but reverse to the channel coding problem where one of the known search algorithms is used to find the path that is closest to the output channel sequence.

II. Sequential Algorithms in Source Coding

The Viterbi algorithm (VA) is known to be the optimal algorithm for trellis search. However, time-space complexity of the VA grows exponentially with the memory length of the decoder so that for the large number of states (e.g. in universal coding) suboptimal algorithms like the M-algorithm, or the stack algorithm have to be used. The Viterbi and the M-algorithm fall into category of breadth-first search algorithms. These techniques extend at once all branches that will ever be extended at the given level. At most M candidate paths are preserved at any time. Another group of search techniques is characterized by extending at any time the path with the best metric – these are metric first algorithms and stack algorithm falls into this class.

Sequential search is a very powerful technique for trellis codes. It has been applied to many problems in recent years and also finds its application in speech coding and recognition. When the search conditions are favorable, Mohan and Anderson [1] discovered that for the same performance stack algorithm is only one half to two thirds as costly as the M-algorithm.

On the other hand, the bidirectional stack algorithm (BSA) is for the first time proposed for decoding convolutional codes in channel coding independently by Kallel and Li [2] and Šenk and Radivojac [3], [4]. Their findings suggest that the Pareto exponent of the unidirectional stack algorithm can be practically doubled by using BSA. In this paper we apply this algorithm to source coding.

Let us first describe the bidirectional procedure. BSA is based on the notions of: 1) reverse trellis code obtained from the original one by time reversing; 2) the tunnel, the unique sequence $0 \le \mathcal{T} \le \mathcal{M}$ branches long that connects two states in the trellis; 3) the tentative decision, the best so far sequence that connects known initial and terminal states; 4) a set of discarding criteria based on the tentative decision aimed to tell beforehand whether a partly explored path is likely to be a part of the finally encoded sequence or not.

BSA uses two stacks: F (forward) and B (backward, used for the reverse code) that can operate almost independently. Its steps are:

BSA1. Put the root node into F stack, and the terminal node into B stack, associating them zero metric. Make one of these stacks active (e.g. the F one);

BSA2. Eliminate the node with the largest metric (of length, say, l) from the active stack. Link it via a tunnel to all the eligible paths from the other stack whose lengths are $L - l + \mathcal{M} - \mathcal{T}$. Store the best path into the tentative decision register. If there is already a path in the register, keep the better. Establish new discarding criteria and discard the paths from both stacks according to them. If both stacks are emptied in this way, the tentative decision is the decoder's final decision. Otherwise, evaluate the metrics of all the successors of the processed path, and eliminate all of them that do not conform to the discarding criteria;

BSA3. Sort the remaining successors into the active stack according to their metrics. Change the active stack and return to step BSA2.

After each tentative decision, a discarding criterion based on non-selection principle [5] is established. This principle states that from two paths diverging from the same node, SA keeps the one whose minimum Fano metric until the end node is maximal. Accumulated distance $d(\cdot)$ and metric $\mu(\cdot)$ are in source coding tied via

$$\mu(\mathbf{v}_{l \cdot n}) = l \cdot n \cdot D^* - d(\mathbf{u}_{l \cdot n}, \mathbf{v}_{l \cdot n}) \qquad (2)$$

where $\mathbf{u}_{ln}$ represents $l \cdot n$ source symbols, $d(\cdot)$ is the distortion incurred in representing the source symbols $\mathbf{u}_{ln}$ by $l \cdot n$ reproduction symbols $\mathbf{v}_{ln}$, and D^* is a bias factor that we call the search bias according to [1]. $10\log_{10}D^*$ is the value of D^* expressed in decibels. Search bias additionally controls the search and enables better comparison between paths at different depths. There is another discarding criterion, based on the finiteness of the stack size. Namely, as soon as a stack (F or B) becomes full, the path with the lowest metric in it is being discarded.

III. Code Design

We have used SGL procedure [6] for finding optimal codes and extending the codebook. This procedure consists of iterative improvement of the codebook by:

1. finding a channel sequence that minimizes distortion between input sequence and decoded sequence using one of the search algorithms described (VA, MA, SA...). This encoding partitions a training sequence such that each partition consists of training samples that were mapped to a particular codeword,
2. given the partition of the training sequence, find a minimum distortion codebook, i.e. calculate new codewords as centroids of each cell.

Steps 1 and 2 are repeated until relative distortion improvement falls below a specified threshold. This training mechanism ensures that given an optimal encoding algorithm new codebook can be no worse than the previous one. It is reasonable to believe that codebooks obtained by this method will correspond to some local optimum even for suboptimal trellis search.

Stewart *et al.* also suggested a method for extension of the decoder. Given a decoder of memory length $\mathcal{M}$, a new decoder of dimension $\mathcal{M} + 1$ can be found by creating a new codebook such that decoder initially gives same values at the output regardless of the value of the channel word that is stored in the extended part of the register. In other words, the algorithm copies the content of the codebook $q - 1$ times into $q^{\mathcal{M}+1}(q - 1)$ allocated positions for new codewords, and then refines them according to above steps.

However, this procedure is intended for and originally applied using optimal search algorithm, i.e. the Viterbi algorithm. This means that, if a search procedure does not perform a full search (of either corresponding trellis or tree), the performance of the code design algorithm can be degraded. In the case of the bidirectional stack algorithm, it is especially true due to the small stack size, so that paths can evade one another seldom forming tentative decisions. This can cause that the best path from either stack reaches the depth $L + \mathcal{M}$ without merging with a path from the opposite stack, making the same decision as the unidirectional SA. In that case the code tends to the one obtained by the unidirectional SA, sometimes forward, sometimes backward. For the purpose of finding codes with good bidirectional properties (i.e. with the best possible bidirectional column distance function) we have modified the SGL algorithm in the following way:

1. find two channel sequences that minimize the distortion between input and destination sequences using unidirectional stack algorithms from both sides (the backward

stack algorithm uses the reverse trellis code and reads a table using reverse states). These two encoded sequences introduce a partition of the training sequence in the same way as in the original SGL algorithm

2. given the partition of the training sequence, find the minimum distortion codebook by calculating reproduction symbols (codewords) as the average values over those elements of the training sequence indexed by the partition cell corresponding to that codewords.

As a consequence of averaging samples from both directions this procedure gives a more balanced code with approximately equal forward and backward column distance profile. The bidirectional stack algorithm is then used as a means for measuring the performance, eventually stopping code design. Once the SNR of the i-th iteration falls below the SNR of the $(i-1)$-th, the code design is stopped. In order to speed up the process, the number of codes generated during the code search procedure is limited.

IV. Coding Speech Sources

In order to compare different algorithms we have chosen to track their computational complexity since memory requirements are quite small. In tree or trellis coding it is convenient to observe only the calls to the copy-decoder (readings from the lookup table). Sometimes, it is reasonable to monitor the inevitable arithmetic operations whose number is proportional to sorting requirements. However, we have measured that such operations are far less time-consuming then the ones corresponding to the copy-decoder calls.

One utterance from the TIMIT database of read speech was used as a training sequence and a second one was used as a test sequence. These recordings were made with sampling frequency of 16 kHz and 16 bits per sample. Total number of samples used for training was 50280, and the number of samples used for testing was 42190. One bit per sample trellis codes ($q = 2$) were generated on training data using the modified SGL procedure proposed in this paper.

Table 1. shows the number of copy-decoder calls and SNR on a test sequence for different stack sizes for decoder memory length $\mathcal{M} = 5$ and $\mathcal{T} = 0$ (there is no gain for higher memory lengths when a single code for all modes of speech stationarity is designed [7]). The data are encoded in 30 ms frames.

TABLE I
SA vs. BSA - performance

	SNR [dB]		Copy-decoder calls	
Stack size	SA	BSA	SA	BSA
2	6.42	7.51	149347	170437
4	9.35	9.85	309901	260876
8	10.26	10.31	774531	479988
16	10.41	10.47	931420	1208543
32	10.62	10.60	2058222	2115267

For $M = 4$ the M-algorithm reaches SNR = 9.45 dB with 337214 copy-decoder calls, and for $M = 8$ SNR = 10.32 dB with 674374.

Figure 3 shows SNR on a training sequence as a function of the memory length, and Fig. 4 SNR obtained on test sequence as a function of computational complexity. The frame length was 20 ms.

We have also tested the unidirectional stack algorithm with codes designed for the bidirectional version and vice versa. The performance of the stack algorithm is ~1 dB worse for bidirectional codes. On the other hand, a bidirectional algorithm always improves performance of its unidirectional counterpart for at least 0.2-0.3 dB for the same or lower complexity. This is due to the two almost independent stacks so that the complexity of the algorithm is the maximum value of those obtained for two directions

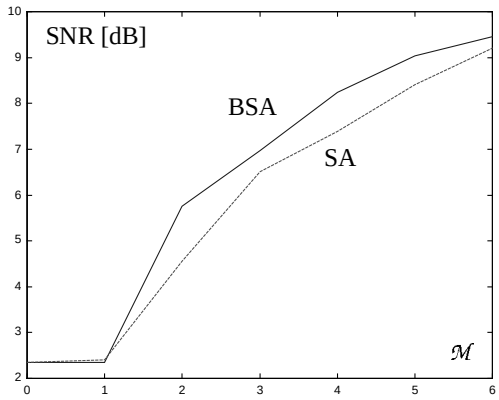

Figure 3. SNR vs. memory length

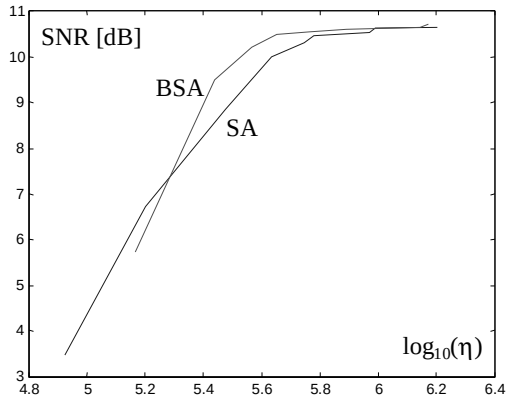

Figure 4. SNR vs. complexity (η)

V. Conclusion

In applications where synchronism is not a crucial factor, such as in stored voice answer-back, metric-first algorithms provide an attractive alternative. In this paper, we have proposed another metric-first algorithm for source coding. How-

ever, unlike in channel coding, improvement of the BSA over SA is not that noteworthy. One of the reasons for that is that this algorithm has to be competitive with the M-algorithm and stack algorithm so that stack size cannot exceed 16 or 32. The overall best path is thus in danger of being discarded from one of the stacks, leaving its counterpart from the other stack the task of recreating the discarded part. In that case, the BSA performs worse than SA in terms of computational complexity, with the same result. If the overall best path is discarded from both stacks, it would also happen for the unidirectional SA, so that no loss is encountered. Another problem is that the bidirectional stack algorithm outperforms the classical SA for longer memories (in channel coding > 30) which allow tunneling thus providing earlier tentative decisions and smaller erasure probability. Unfortunately, the unsegmented speech signal is not so highly correlated to enable using memories greater than 5-6 or 10-12 in universal (segmented) applications. Finally, it is difficult to say whether good covering codes with good bidirectional column distance profile exist, as do packing codes needed for channel coding.

REFERENCES

[1] S. Mohan and J. B. Anderson, "Speech Encoding by a Stack Algorithm," *IEEE Transactions on Communications*, VOL. COM-28, pp. 825-830, No. 6, June 1980.

[2] S. Kallel and Kaiping Li, "Bidirectional Sequential Decoding," *IEEE Transactions on Information Theory*, vol. IT-43, No. 4, pp. 1319-1326, July 1997.

[3] V. Šenk and P. Radivojac, "The Bidirectional Stack Algorithm - Simulation Results," Proceedings of TELSIKS'95, 2-nd Conference on Telecommunications in Modern Satellite and Cable Services, pp. 349-352, Niš, Yugoslavia, October 1995.

[4] V. Šenk and P. Radivojac, "The Bidirectional Stack Algorithm," Proceedings of the 1997 IEEE International Symposium on Information Theory, ISIT'97, p. 500, Ulm, Germany, July 1997.

[5] 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.

[6] L. C. Stewart, R. M. Gray, and Y. Linde, "The Design of Trellis Waveform Coders," *IEEE Transactions on Communications*, vol. COM-30, No. 4, pp. 702-710, April 1982.

[7] V. Šenk and P. Radivojac, "Universal Trellis Coding of Speech," Proceedings of the International Conference on Telecommunications, ICT'98, Vol. II, pp. 273-277, Porto Carras, Greece, June 1998.