

A novel turbo-TCM scheme based on concatenated tree codes

Baoming Bai[†], K. S. Ho^{††}, and Li Ping^{††}

Summary

In this letter, we introduce a two-state turbo-TCM scheme based on the concatenated tree codes. The proposed scheme can achieve near capacity performance yet has considerably lower decoding complexity compared with other existing turbo-TCM codes.

Key words:

Error-correcting codes, parallel concatenated codes, turbo codes, trellis-coded modulation.

1. Introduction

Various approaches to turbo-type [1] concatenated trellis-coded modulation (T-TCM) have been investigated [2][3]. In this letter, we present a new class of concatenated two-state TCM (CT-TCM) codes based on the concatenated tree codes [4]. The proposed scheme can achieve near capacity performance yet has considerably lower decoding complexity compared with other existing T-TCM codes.

2. Tree Codes and Two-State Trellis

Tree codes are introduced in [4]. As examples, the encoding principles of two different tree codes are shown in Fig. 1 using Bayesian networks. Here, every node represents a bit in the code, distinguished by a white one (such as $d_{k,0}$) for an information bit and a black one (such as q_k) for a parity bit. The latter is the parity check of the starting nodes of its incoming edges. Denote the k th information symbol by $\mathbf{d}_k = (d_{k,0}, d_{k,1})$. The corresponding coded symbol is given by (\mathbf{d}_k, q_k) .

Tree codes can also be represented by two-state trellises with parallel branches, Fig. 1. Here the two state variables at time k are the two valid values of the parity bit q_k . Branches in the trellis are labeled by the values of (\mathbf{d}_k, q_k) . Notice that $d_{k,1}$ in Fig. 1(b) does not participate in parity check, which results in different branch connections in Fig. 1(b) as opposed to Fig. 1(a).

Parallel branches are usually avoided in conventional coding schemes for performance concerns. However, in the two trellises in Fig. 1, a pair of parallel branches in one code will not be parallel to each other in another code. (For example, (11,0) and (00,0) represent two parallel branches in Fig. 1(a) but not in Fig. 1(b).) This property is

crucial for overcoming the parallel transition problem after concatenating several such component codes together.

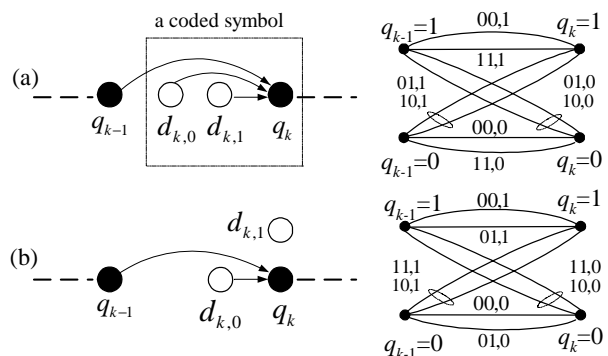


Fig. 1. Two examples of tree codes and their trellis representations. White and black circles represent information and parity bits, respectively.

3. Concatenated Two-State TCM Codes

3.1 Two-State TCM Encoder

A two-state TCM encoder consists of a binary tree encoder followed by a signal mapper. Based on Ungerboeck set partition rule, the labels of the trellis branches (i.e., the coded symbols) in Fig. 1 are one-to-one mapped to a multi-ary constellation.

3.2 Concatenated Two-State TCM (CT-TCM) Codes

In the proposed scheme, multiple two-state TCM codes are concatenated in a turbo-type manner. Similar to [2], symbol interleavers are used. Puncturing is used to improve the spectral efficiency. We will always assume that one and only one of signals carrying the same information symbol is transmitted.

3.3 Encoder Structure Ignoring Interleavers

As a design example, Fig. 2 shows the encoder structure employing 8-PSK with a throughput of 2bits per signal. We first ignore any interleaving operation. Let M be the number of component codes, and n be the number of information bits contained in an input symbol. It is easy to verify that the code shown in Fig. 2 has the following properties.

[†] The author is with National Key Lab. of ISN, Xidian University, Xi'an 710071, China.

^{††} The authors are with Department of Electronic Engineering, City University of Hong Kong, Kowloon, Hong Kong. (email: eeliping@cityu.edu.hk)

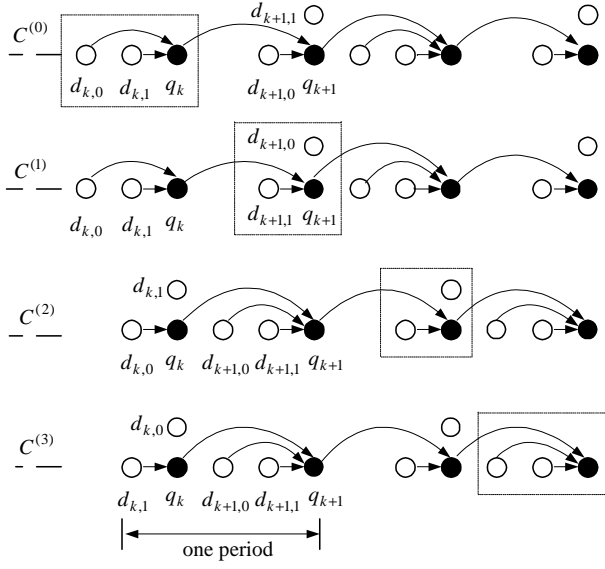


Fig. 2. Structure of the CT-TCM encoder for 8-PSK. The symbols inside the dashed block are transmitted and other symbols are punctured. “ $C^{(m)}$ ” represents the m th component code.

Property 1: With $M \geq n+1$ and $M > 2$, any nonzero information symbol d_k may result in cross transitions during encoding, i.e., either ($q_{k-1} = 0 \rightarrow q_k = 1$) or ($q_{k-1} = 1 \rightarrow q_k = 0$), in at least two component codes.

Property 2: Consider two information words \mathbf{a} and \mathbf{b} that differ in only one symbol d_k . Due to Property 1, in at least two component codes, the encoding paths of \mathbf{a} and \mathbf{b} will split at d_k and then remain separated afterwards. (This is a kind of recursive property similar to that of recursive convolutional codes [1].)

Property 3: Consider two information words \mathbf{a} and \mathbf{b} that differ in only two symbols d_k and $d_{k'}$, assuming $k < k'$. Due to Properties 1 and 2, in at least two component codes, the encoding paths of \mathbf{a} and \mathbf{b} will split at d_k and then remain separated afterwards until $d_{k'}$. They may or may not remerge at $d_{k'}$.

3.4 Impact of Interleavers

We adopt symbol-based interleavers since they result in lower decoding complexity and better waterfall behavior [2]. A modulo- M symbol interleaver $\pi(i)$ is constrained by

$$\pi(i) \bmod M = i \bmod M, \quad i = 0, 1, \dots, N-1. \quad (1)$$

Such interleavers are used in conjunction with the following modulo- M puncturing rule. For the m th

component code, we puncture all the modulated symbols except those at position $\{i \mid i \bmod M = m\}$. (Refer to Fig. 2.) When $M=2$, the above modulo- M interleaving-puncturing rule is equivalent to the odd-even interleaving scheme used in [2].

For the code in Fig. 2, with the modulo-4 interleavers, the three properties mentioned above still apply except that the splitting and remerging positions may be different in different component codes. The non-overlapping spans of the encoding paths of \mathbf{a} and \mathbf{b} are thus random variables. Such an effect is equivalent to the interleaving gain for turbo codes, which results in the reduction in the number of nearest neighboring codewords. This argument can be generalized to codewords generated by information words that differ in more than two symbols.

4. Complexity of the Decoder

The CT-TCM decoder structure is based on the multi-dimensional turbo decoder incorporating the BCJR algorithm, as detailed in [4]. We now show that CT-TCM codes have very low decoding complexities.

4.1 BCJR Algorithm for Component Codes

Consider the k th section in a trellis diagram. Let s_k be the encoder state at time k and d_k be the information symbol associated with the state transition $s_{k-1} \rightarrow s_k$. (Note: For CT-TCM, $s_k = q_k$, see Fig. 1.) The corresponding modulated symbol is designated by x_k . Let $\mathbf{y} = \{y_k\}$ be the noisy observation of $\mathbf{x} = \{x_k\}$, and $P(\mathbf{x}_k | \mathbf{y}_k) = P(d_k, s_{k-1} \rightarrow s_k | y_k)$ be the branch metric of the transition $s_{k-1} \rightarrow s_k$ caused by d_k . Denote by $B(s_{k-1}, s_k)$ the set of all the parallel branches between s_{k-1} and s_k . The BCJR algorithm is summarized as follows [1].

Stage 1: Preparation: Finding the combined branch metric between s_{k-1} and s_k ,

$$\gamma(s_{k-1}, s_k) = \sum_{B(s_{k-1}, s_k)} P(\mathbf{x}_k | \mathbf{y}_k) \quad (2)$$

Stage 2: Forward recursion:

$$\alpha(s_k) = \sum_{s_{k-1}} \gamma(s_{k-1}, s_k) \alpha(s_{k-1}) \quad (3)$$

Stage 3: Backward recursion:

$$\beta(s_{k-1}) = \sum_{s_k} \gamma(s_{k-1}, s_k) \beta(s_k) \quad (4)$$

Stage 4: Output:

$$P(d_k = i | \mathbf{y}) \propto \sum_{x_k: d_k=i} P(\mathbf{x}_k | \mathbf{y}_k) \alpha(s_{k-1}) \beta(s_k) \quad (5)$$

4.2 Complexity Analysis

Again, let M be the number of component codes, S be the number of states of each component code and n be the number of information bits contained in an input symbol.

Normalizing $\alpha(s_k = 0)$ and $\beta(s_k = 0)$ to 1 for every k can reduce decoding cost considerably for trellises with $S=2$. For example, for $\alpha(0) = \beta(0) = 1$, it is not necessary to carry out the associated multiplication inside the summation in (3)-(5).

The above cost saving also applies to trellises with $S \geq 4$, but the relative benefit reduces as the number of states increases. Table 1 provides a complexity comparison involved in (2)-(5) for a 2-state and an S -state ($S \geq 4$) code respectively. The normalization costs are also included in Table 1, which are S additions and S divisions in Stage 2 and 3, respectively.

Table 1. The computational cost of the BCJR algorithm (ADD for addition and MUL for multiplication)

operation	Case-I (S -state)		Case-II (2-state)	
	ADD	MUL	ADD	MUL
Stage 1	0	0	$2^n - 2$	0
Stage 2	$(2^n - 1)S + S$	$2^n S + S$	2	3
Stage 3	$(2^n - 1)S + S$	$2^n S + S$	2	3
Stage 4	$2^n S$	$2^n S$	2^n	$\frac{3}{2} \times 2^n$
Total	$(3 \times 2^n)S$	$(3 \times 2^n)S + 2S$	$2^{n+1} + 2$	$\frac{3}{2} \times 2^n + 6$

It is seen from Table 1 that the decoding cost of a 2-state trellis code is about $1.5S \sim 2S$ times lower than that of an S -state ($S \geq 4$) trellis code without parallel branches. This ratio should be adjusted considering the number of component codes. For example, the complexity of the CT-TCM code in Fig. 2 ($M=4$) is about six times lower than that of the 8-state, $M=2$, turbo-TCM code in [2], and twelve times lower than that of the 16-state, $M=2$, turbo-TCM code in [3].

5. Numeric Results

Fig. 3 illustrates the performance of the CT-TCM code in Fig. 2. The results of [2] are also plotted in Fig. 3 for reference. It is seen that the proposed scheme has performance comparable to that presented in [2]. However, the proposed scheme has considerably lower complexity.

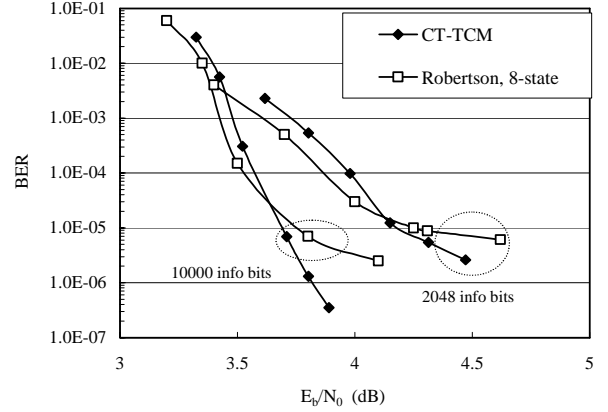


Fig. 3. BER performance of the CT-TCM code for 8-PSK, 2bits/symbol.

6. Conclusions

By combining the concatenated tree codes with TCM, we have proposed a very low complexity, high performance CT-TCM scheme for bandwidth efficient modulations.

Acknowledgments

The authors wish to thank the anonymous reviewers for their helpful suggestions and comments. This work was supported by CERG under grand N_CityU N_101/01 and NSFC under Grant 60131160742.

References

- [1] C. Berrou, A. Glavieux, "Near optimum error correcting coding and decoding: Turbo-codes," IEEE Trans. Commun., vol.44, no.10, pp.1261-1271, Oct. 1996.
- [2] P. Robertson and T. Worz, "Bandwidth-efficient turbo trellis-coded modulation using punctured component codes," IEEE J. Select. Areas Commun., vol.16, no.2, pp.206-218, Feb. 1998.
- [3] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Parallel concatenated trellis coded modulation," Proc. IEEE ICC'96, pp.974-978, 1996.
- [4] Li Ping and K.Y. Wu, "Concatenated tree codes: A low complexity, high performance approach," IEEE Trans. Inform. Theory, vol.47, no.2, pp.791-799, Feb. 2001.