

# An Improved Two-State Turbo-SPC Code for Wireless Communication Systems

Keying Wu, *Student Member, IEEE*, and Li Ping, *Member, IEEE*

**Abstract**—This letter presents an improved two-state turbo single-parity-check code for applications with rates around 1/3. The new code is compared with the  $(15, 13)_8$  turbo code used in the Third Generation Partnership Project. Simulation results demonstrate that the proposed code can achieve similar performance at reduced decoding complexity.

**Index Terms**—A posteriori probability (APP) decoding, iterative decoding, turbo codes, union bounds.

## I. INTRODUCTION

**T**URBO CODES are a family of powerful error-correcting codes [1], [2]. Recently, it has been shown that the concatenated tree (CT) codes [3] provide a low-complexity alternative to turbo codes without compromising performance. Some good CT codes can also be regarded as special two-state turbo single-parity-check (SPC) codes [4]. For rates  $R \geq 1/2$  and nearly all code lengths, CT codes can achieve similar performance as turbo codes. However, CT codes generally have much lower decoding complexities than turbo codes.

We observed that CT codes work best for rates  $R \geq 1/2$ . For rates lower than 1/2, the relative gap between the simulated performance and the corresponding Shannon limit increases. Recall that codes with lower rates are expected to work in noisier channels [i.e., with lower signal-to-noise ratios (SNRs)]. It appears that stronger constituent codes (than tree codes – the constituent codes of CT codes) are required in very noisy channels. A solution for very low rates (say,  $R \leq 1/8$ ) has been devised in [5] based on Hadamard codes. The problem can also be treated using turbo-SPC codes [4] or multiple turbo codes [6] with higher state numbers. (Turbo-SPC codes and multiple turbo codes with two states are actually CT codes.) However, all these options incur increased decoding costs.

In this letter, we present a technique to improve the performance of two-state turbo-SPC codes, while maintaining their low-cost property. The basic principle is to employ stronger two-state constituent codes. Since the state number is kept at two, the normalized decoding complexity of the improved code is still very low. The advantage of the improved code is most noticeable in the range of  $1/8 < R < 1/2$ . For  $R \leq 1/8$ , the turbo-Hadamard codes discussed in [5] can lead to better performance. For  $R \geq 1/2$ , the CT codes discussed in [3] are sufficiently good.

Paper approved by E. Ayanoglu, the Editor for Communication Theory and Coding Applications of the IEEE Communications Society. Manuscript received December 21, 2001; revised July 29, 2002. This work was supported by a grant from the Research Grant Council of the Hong Kong SAR under Grant CityU 1010/01E.

The authors are with the Department of Electronic Engineering, City University of Hong Kong, Kowloon, Hong Kong (e-mail: kewu@ee.cityu.edu.hk; eeliping@cityu.edu.hk).

Digital Object Identifier 10.1109/TCOMM.2004.833042

An interesting application of the improved code is in wireless communication systems. The rate-1/3  $(15, 13)_8$  turbo code has been adopted for channel protection in the Third Generation Partnership Project (3GPP). This is supported by the fact [7] that, for modest code lengths (say, interleaver length  $N \leq 5000$ ) and comparable complexity, this code can provide better performance, compared with other candidates, such as low-density parity-check (LDPC) codes [8] and other turbo-like codes [9], [10]. However, this letter shows that the improved two-state turbo-SPC code can be a more efficient solution. In particular, we will show that the decoding cost of the rate-1/3 improved turbo-SPC code is only a fraction of that of the  $(15, 13)_8$  turbo code, while the performance of the two codes is very similar.

## II. ENCODING AND DECODING PRINCIPLES

### A. Rate-2/4 Systematic Recursive Convolutional Code

Fig. 1(a) is the trellis diagram of a standard rate-1/2, two-state systematic recursive convolutional code. The first bit in the branch output is the systematic bit, and the second bit is the parity bit. Fig. 1(b) is the trellis diagram of a rate-2/4 two-state systematic recursive convolutional code. The first two bits in the branch output are systematic bits, and the last two are parity bits.

The code in Fig. 1(a) has a minimum distance of three, and the code in Fig. 1(b) has a minimum distance of four. Therefore, the latter is stronger than the former, and is more suitable to work in relatively noisy channels.

### B. SPC Precoding Technique

We adopt the SPC precoding technique introduced in [4] for rate adjustment. The basic principle is as follows. The information sequence is arranged into an array  $\mathbf{D} = \{\mathbf{d}_k\}$  with  $J$  rows and  $K$  columns, where  $\mathbf{d}_k = [d_{k,1} \cdots d_{k,j} \cdots d_{k,J}]^T$  is the  $k$ th column vector. In the encoding process, we first generate  $\mathbf{q} = \{q_k\}$ , where  $q_k$  is the parity check of  $\mathbf{d}_k$ . Then  $\mathbf{q}$  is encoded using a systematic code  $\hat{C}$ , producing a parity vector  $\mathbf{p}$ . We use  $\{\mathbf{D}, \mathbf{p}\}$  to form a new code  $C$ . (Note: The intermediate vector  $\mathbf{q}$  does not appear in  $C$ .) If  $\hat{C}$  has a rate of 1/2 (or 2/4), the rate of  $C$  is raised to

$$R = \frac{J}{J+1}. \quad (1)$$

If we choose the rate-1/2 convolutional code in Fig. 1(a) as  $\hat{C}$ , the resultant code is a tree code (the constituent code of the CT code) [3]. From our observations, tree codes are not strong enough to construct concatenated codes with rates lower than 1/2. To solve this problem, we employ the rate-2/4 convolutional code in Fig. 1(b) as  $\hat{C}$  in this letter. Since the code in Fig. 1(b) has a larger minimum distance than the code in Fig. 1(a), the resultant constituent codes are stronger than tree codes, and more suitable for low-rate applications where channels are very noisy.

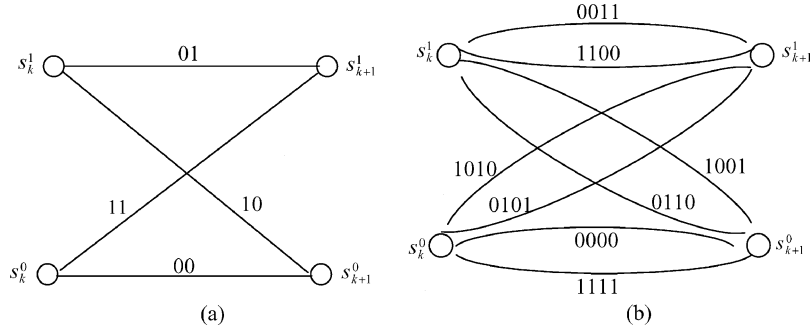


Fig. 1. Trellis diagrams of two-state systematic recursive convolutional codes with rates (a) 1/2 and (b) 2/4.

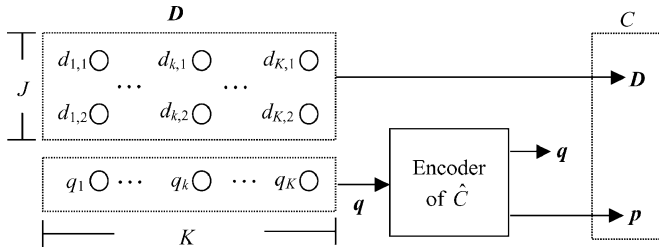
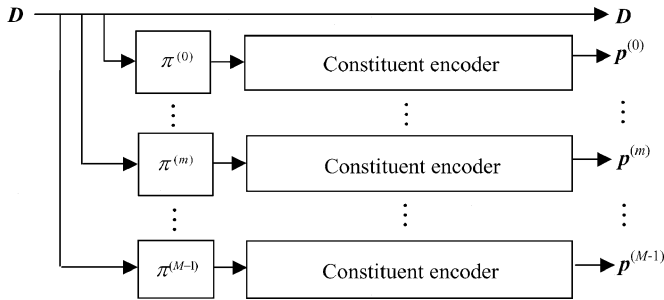

 Fig. 2. Encoder of the proposed constituent code (with  $J = 2$ ).

 Fig. 3. Encoder of the improved two-state turbo-SPC code, where  $\pi^{(m)}$  is the interleaver for the  $m$ th constituent code. All the constituent encoders have the structure shown in Fig. 2.

Fig. 2 shows the corresponding encoder structure with  $J = 2$ . The trellis representation of  $C$  can be derived directly from that of  $\hat{C}$  [4]. Based on its trellis representation, we can obtain the weight-enumerating function (WEF) of  $C$  using the technique introduced in [11], which is useful in calculating the union bound [2] for the overall turbo-SPC code (as will be discussed in Section II-C).

### C. The Global Code

We construct an improved two-state turbo-SPC code by concatenating  $M$  codes in Fig. 2 in parallel via interleavers, as shown in Fig. 3. The overall codeword is formed by  $\{D, \mathbf{p}^{(0)}, \dots, \mathbf{p}^{(m)}, \dots, \mathbf{p}^{(M-1)}\}$ , where  $\mathbf{p}^{(m)}$  is the parity sequence generated by the  $m$ th constituent encoder. The overall turbo-SPC code has a rate

$$R = \frac{J}{J + M}. \quad (2)$$

Assuming uniform interleavers, the WEF of the improved turbo-SPC code can be derived directly from those of its constituent codes, using the technique introduced in [2]. From the

WEF, we can calculate the union bound on bit-error rate (BER) for the improved turbo-SPC code as [2]

$$P_b \leq \frac{1}{2} \sum_d \sum_{\substack{w,j \\ w+j=d}} \frac{w}{N} A_{w,j} \operatorname{erfc} \left( \sqrt{d \frac{E_b R}{N_0}} \right) \quad (3)$$

where  $N$  is the interleaver length, and  $A_{w,j}$  the number of codewords with information weight  $w$  and parity weight  $j$ .

### D. Decoding Principle

The decoding procedure of the proposed code includes the local decoding of constituent codes and the global decoding of the overall code. With the trellis representation of the constituent code, the local decoding can be carried out using the Bahl-Cocke-Jelinek-Raviv (BCJR)-based algorithm in [4]. The global decoding can be accomplished using the serial turbo-type decoder [3]. We will explain the decoding principle in more detail in Section III-B, using a specific rate-1/3 improved turbo-SPC code as an example.

## III. EXAMPLE OF RATE-1/3 IMPROVED TWO-STATE TURBO-SPC CODE

### A. Encoding Procedure

Return to Fig. 2. Set  $J = 2$  and choose the code in Fig. 1(b) as  $\hat{C}$ . Using the resultant code as constituent codes and setting  $M = 4$ , we form a rate-1/3 improved two-state turbo-SPC code. Denote by  $D = \{\mathbf{d}_k\}$  the  $2 \times K$  information matrix, where  $\mathbf{d}_k = [d_{k,1} d_{k,2}]^T$ . The encoding procedure of the constituent code is summarized as follows.

- Step 1) For every  $k$ , calculate  $q_k = d_{k,1} + d_{k,2} \bmod 2$ .
- Step 2) Use  $\mathbf{q} = \{q_k\}$  to drive  $\hat{C}$ , producing a parity vector  $\mathbf{p} = \{p_k\}$ .
- Step 3) The final output is  $\{D, \mathbf{p}\}$ .

### B. Decoding Procedure

Here we only consider the local decoding of constituent codes. The global decoding can be accomplished using the serial turbo-type decoder discussed in [3].

Assume that  $\mathbf{c} = \{c_i\}$  is the transmitted codeword in the binary phase-shift keying (BPSK) format  $\{0 \leftrightarrow +1, 1 \leftrightarrow -1\}$ . Denote by  $\text{obs}(\mathbf{c})$  the noisy observation of  $\mathbf{c}$ . The *a posteriori*

probability (APP) decoding is to evaluate the *a posteriori* likelihood ratios (LRs)

$$R(c_i|\text{obs}(\mathbf{c})) \equiv \frac{\Pr(c_i = +1|\text{obs}(\mathbf{c}))}{\Pr(c_i = -1|\text{obs}(\mathbf{c}))}, \quad i = 1, 2, \dots \quad (4)$$

We define the following LR value of  $c_i$  that is calculated from its channel output  $\text{obs}(c_i)$ :

$$\tilde{R}(c_i) \equiv \frac{\Pr(c_i = +1|\text{obs}(c_i))}{\Pr(c_i = -1|\text{obs}(c_i))}, \quad i = 1, 2, \dots \quad (5a)$$

Assuming that there is no *a priori* information about  $c_i$ ,  $\tilde{R}(c_i)$  can be calculated as

$$\tilde{R}(c_i) = \frac{p(\text{obs}(c_i)|c_i = +1)}{p(\text{obs}(c_i)|c_i = -1)}, \quad i = 1, 2, \dots \quad (5b)$$

Similarly, we also write  $\tilde{R}(d_{k,j})$  and  $\tilde{R}(p_k)$  depending on whether  $c_i$  is an information or parity bit. Assume that  $\{\tilde{R}(d_{k,j})\}$  and  $\{\tilde{R}(p_k)\}$  are available. The decoding procedure can be carried out in three steps.

Step 1) For every  $k$ , calculate [4]

$$\tilde{E}(q_k) \equiv \frac{\Pr(q_k = +1|\text{obs}(d_{k,1}), \text{obs}(d_{k,2}))}{\Pr(q_k = -1|\text{obs}(d_{k,1}), \text{obs}(d_{k,2}))}. \quad (6a)$$

Recall that  $q_k$  is the parity check of  $d_{k,1}$  and  $d_{k,2}$ . From [4],  $\tilde{E}(q_k)$  can be calculated as

$$\tilde{E}(q_k) = f(\tilde{R}(d_{k,1}), \tilde{R}(d_{k,2})) \quad (6b)$$

where  $f(x, y) = (xy + 1)/(x + y)$  [4]. (Cost: 1 addition and 1 multiplication per information bit, per constituent code.)

Step 2) Decode  $\hat{C}$  using  $\{\tilde{E}(q_k)\}$  and  $\{\tilde{R}(p_k)\}$  as the input, producing the extrinsic LR of  $\mathbf{q}$ , denoted by  $\{E(q_k)\}$ .<sup>1</sup> This procedure can be accomplished by the BCJR algorithm [1]. (Cost: 5 additions and 5.5 multiplications per information bit, per constituent code.)

Step 3) For each  $d_{k,j}$ , calculate its *a posteriori* LR in (4) as [4]

$$R(d_{k,1}|\text{obs}(\mathbf{c})) = \tilde{R}(d_{k,1}) \cdot f(\tilde{R}(d_{k,2}), E(q_k)) \quad (7a)$$

$$R(d_{k,2}|\text{obs}(\mathbf{c})) = \tilde{R}(d_{k,2}) \cdot f(\tilde{R}(d_{k,1}), E(q_k)). \quad (7b)$$

Here, the  $f$ -function is defined in the same way as in Step 1, since we are still dealing with an SPC constraint. (Cost: 2 additions and 3 multiplications per information bit, per constituent code.)

We briefly explain an advantage of two-state trellis codes when the BCJR algorithm is applied. During the forward and backward operations in the BCJR algorithm, we can normalize one state metric at each depth to one, and then the multiplications associated with this state are not necessary. Such cost saving is considerable for a two-state trellis (50% of the multiplications), but less impressive for trellises with more than two states (e.g., 12.5% of the multiplications for an eight-state trellis). Therefore, the decoding cost of a two-state trellis code is close to 1/8 (or more precisely, much less than 1/4) of that of an eight-state one. Although the trellis in Fig. 1(b) is more complex than the one in Fig. 1(a), the normalized decoding costs per information bit are very similar for the two codes, since the former carries two information bits per trellis branch, while the latter carries only one.

<sup>1</sup>Note that  $\tilde{E}(q_k)$  and  $E(q_k)$  are both extrinsic LR about  $q_k$ .  $\tilde{E}(q_k)$  is based on the local SPC constraint, and  $E(q_k)$  is based on the constraint of  $\hat{C}$ .

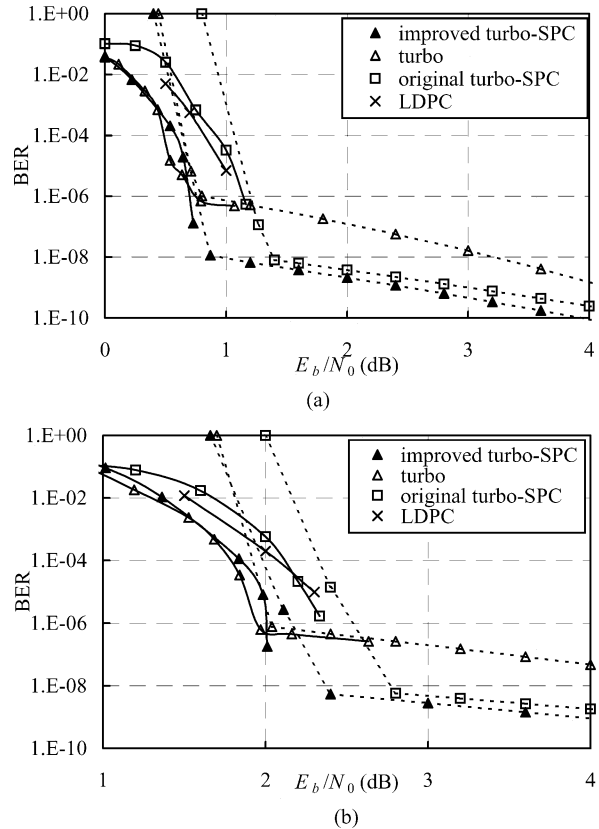


Fig. 4. Performance comparison among the rate-1/3 improved turbo-SPC code and some other rate-1/3 codes in (a) AWGN channels and (b) fully interleaved Rayleigh fading channels. All codes (except the LDPC codes) have information length = 2896 and 18 decoding iterations. Both improved turbo-SPC and CT codes have  $J = 2$  and  $M = 4$ , which are based on the codes in Fig. 1(b) and (a), respectively. Results for the LDPC codes are cited from [12] with information length = 3072. Solid lines are for simulation results and dashed lines for bound results.

### C. Comparisons

Fig. 4(a) and (b) show the performance of the rate-1/3 improved two-state turbo-SPC code in additive white Gaussian noise (AWGN) and fully interleaved Rayleigh fading channels. For comparison, we also include the performance of the  $(15, 13)_8$  turbo code used in 3GPP, two LDPC codes [12], and a CT code [3] (i.e., an original two-state turbo-SPC code [4]). The two LDPC codes from [12] are optimized for AWGN and fully interleaved Rayleigh fading channels, respectively. All codes have  $R = 1/3$ . The performance of the proposed code is similar to that of the turbo code, and superior to that of the LDPC and CT codes in the BER range above  $10^{-6}$ .

For the BER range below  $10^{-6}$ , simulation becomes difficult. With random interleavers, the union bound results can be used instead, which are included for all the codes in Fig. 4 except the LDPC codes. It is interesting to see that the proposed turbo-SPC code has lower error floors than the turbo code with random interleavers. Optimized interleavers, such as the  $S$ -random interleaver [13], may potentially be used to improve the error-floor performance of both codes. However, interleaver optimization is a complicated issue for the proposed code, since it involves multiple interleavers. We are still investigating this issue.

Table I shows the decoding costs of the codes in Fig. 4. The decoding complexity of an  $S$ -state turbo code can be estimated,

TABLE I  
COMPLEXITY/STORAGE REQUIREMENT COMPARISON FOR CODES IN FIG. 4

Complexity/storage requirement (per information bit)	Improved turbo-SPC	Turbo	LDPC	CT
Additions (per iteration)	32	88	105	40
Multiplications (per iteration)	38	130	126	48
Interleaving index	3	1	21	3

based on the BCJR algorithm listed in [14] as (per trellis section per iteration per constituent code):  $2S - 1$  additions and  $3S$  multiplications for the forward search;  $2S - 1$  additions and  $3S$  multiplications for the backward search;  $2(S - 1)$  additions and  $2S + 1$  multiplications for the output stage. The two LDPC codes have an average variable degree of  $t \approx 7$ , and the CT code can be regarded as a special LDPC code with  $t \approx 8/3$  [3]. Their decoding costs can be estimated as  $5t/R$  additions and  $6t/R$  multiplications per information bit, per iteration [3]. Clearly, the improved turbo-SPC code has the lowest decoding complexity among the codes compared.

Table I also shows the storage requirements of interleavers for the codes in Fig. 4. For the turbo code, one index per information bit is required. (Note: The first constituent code does not need an interleaver.) For the turbo-SPC and CT codes, three indexes per information bit are required. For the LDPC codes,  $t/R \approx 21$  indexes per information bit are required. (For each variable node in an LDPC code, its connections to different check nodes should be stored. Thus, the total storage requirement is  $t \times L$  indexes, where  $L$  is the total number of variable nodes, and the normalized storage requirement is  $t/R$  indexes per information bit.) In this case, the turbo code has an advantage.

It can also be shown that other options, such as those in [6] and [7], may offer similar performance to the turbo code and proposed code. However, the decoding complexities of these alternatives are generally higher than that of the proposed code.

Here, we briefly discuss the impact of  $J$  and  $M$  on the performance and decoding complexity of the improved turbo-SPC code. Fixing  $R$ ,  $M$  increases with  $J$  [see (2)]. We observed that both the waterfall and error-floor behaviors of the proposed code improve as  $M$  increases up to four. Then the waterfall performance begins to decline, and the error-floor performance continues to improve. The decoding complexity of the proposed code also increases with  $M$ .

## IV. CONCLUSIONS

We have proposed a technique to improve the two-state turbo-SPC codes for applications with rates around 1/3. The high-performance and low-complexity properties of the rate-1/3 proposed code are illustrated in Fig. 4 and Table I, compared with the  $(15, 13)_8$  turbo code used in 3GPP and some other rate-1/3 coding schemes. This demonstrates the potential application of the proposed code in future wireless communication systems.

## REFERENCES

- [1] C. Berrou and A. Glavieux, "Near-Shannon-limit error-correcting coding and decoding: Turbo codes," *IEEE Trans. Commun.*, vol. 44, pp. 1261–1271, Oct. 1996.
- [2] S. Benedetto, "Unveiling turbo codes: Some results on parallel concatenated coding schemes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 409–428, Mar. 1996.
- [3] L. Ping and K. Y. Wu, "Concatenated tree codes: A low-complexity, high-performance approach," *IEEE Trans. Inform. Theory*, vol. 47, pp. 791–799, Feb. 2001.
- [4] L. Ping, "Turbo-SPC codes," *IEEE Trans. Commun.*, vol. 49, pp. 754–759, May 2001.
- [5] L. Ping, W. K. Leung, and K. Y. Wu, "Low-rate turbo-Hadamard codes," *IEEE Trans. Inform. Theory*, vol. 49, pp. 3213–3224, Dec. 2003.
- [6] D. Divsalar and F. Pollara, "Multiple turbo codes," in *Proc. MILCOM*, vol. 1, 1995, pp. 279–285.
- [7] S. Dolinar, D. Divsalar, A. Kiely, and F. Pollara, "Performance-complexity tradeoffs for turbo and turbo-like codes," in *Proc. IEEE Int. Symp. Information Theory*, Washington, DC, 2001, p. 101.
- [8] R. M. Tanner, "A recursive approach to low-complexity codes," *IEEE Trans. Inform. Theory*, vol. IT-27, pp. 533–547, Sept. 1981.
- [9] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Analysis, design and iterative decoding of double serially concatenated codes with interleavers," *IEEE J. Select. Areas Commun.*, vol. 16, pp. 231–245, Feb. 1998.
- [10] H. Jin and R. J. McEliece, "Coding theorems for turbo code ensembles," *IEEE Trans. Inform. Theory*, vol. 48, pp. 1451–1461, June 2002.
- [11] A. M. Viterbi, A. J. Viterbi, J. Nicolas, and N. T. Sindhusayana, "Perspectives on interleaved concatenated codes with iterative soft output decoding," in *Proc. IEEE Int. Symp. Turbo Codes*, Brest, France, Sept. 1997, pp. 47–54.
- [12] J. Hou, P. H. Siegel, and L. B. Milstein, "Performance analysis and code optimization of low-density parity-check codes on Rayleigh fading channels," *IEEE J. Select. Areas Commun.*, vol. 19, pp. 924–934, May 2001.
- [13] D. Divsalar and F. Pollara, "Turbo codes for PCS applications," in *Proc. IEEE Int. Conf. Communications*, vol. 1, June 1995, pp. 54–59.
- [14] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and suboptimal MAP decoding algorithms operating in the log domain," in *Proc. IEEE Int. Conf. Communications*, vol. 2, June 1995, pp. 1009–1013.