Obtaining Extra Coding Gain for Short Codes by Block Markov Superposition Transmission

Xiao Ma, Chulong Liang, Kechao Huang, and Qiutao Zhuang

Department of Electronics and Communication Engineering
Sun Yat-sen University
Guangzhou 510006, GD, China
Email: maxiao@mail.sysu.edu.cn

***Abstract*—In this paper, we present a new approach, called block Markov superposition transmission (BMST), to construct from short codes a class of convolutional codes with large constraint length. The BMST is very similar to superposition block Markov encoding (SBME), which has been widely used to prove multiuser coding theorems. We also present an iterative sliding-window decoding algorithm for the proposed transmission scheme. The extra coding gain obtained by BMST can be bounded in terms of the Markov order and with the help of the input-output weight enumerating function (IOWEF) of the BMST system, which can be computed from that of the short code by performing a trellis-based algorithm. Numerical results verify our analysis and show that an extra coding gain of 6.4 dB at bit-error rate (BER) 10^{-5} can be obtained by BMST of the $[7,4]$ Hamming code.**

I. Introduction

It is an old subject to construct long codes from short codes [1]. Here, by *short codes*, we mean block codes with short code lengths or convolutional codes with short constraint lengths. Product codes [2], presented by Elias in 1954, may be the earliest method for constructing long codes from short codes. An $[n_1n_2, k_1k_2]$ product code is formed by an $[n_1, k_1]$ linear code $\mathscr{C}_1$ and an $[n_2, k_2]$ linear code $\mathscr{C}_2$. Each codeword of the product code is a rectangular array of n_1 columns and n_2 rows in which each row is a codeword in $\mathscr{C}_1$ and each column is a codeword in $\mathscr{C}_2$. In 1966, Forney [3] proposed a class of codes, called concatenated codes. Typically, a concatenated code investigated by Forney consists of a relatively short code as an inner code and a relatively long algebraic code as an outer code. In 1993, Berrou *et al* [4] invented turbo codes, by which researchers have been motivated to construct capacity-approaching codes. The original turbo code [4] consists of two convolutional codes which are parallelly concatenated by a pseudo-random interleaver, and hence is also known as a parallel concatenated convolutional code (PCCC) [5]. Since the invention of turbo codes, concatenations of simple interleaved codes have been proved to be a powerful approach to design iteratively decodable capacity-approaching codes [6–10]. Another class of capacity-approaching codes, namely, low-density parity-check (LDPC) codes, which were proposed in the early 1960s and rediscovered after the invention of turbo codes, can also be considered as concatenations of interleaved single parity-check codes and repeat codes [11–17].

In this paper, we present a new approach, called block Markov superposition transmission (BMST), to construct long codes from short codes. The BMST is very similar to superposition block Markov encoding (SBME), which has been widely used to prove multiuser coding theorems. The method of SBME was first introduced for the multiple-access channel with feedback by Cover and Leung [18] and successfully applied by Cover and El Gamal [19] for the relay channel. The idea behind SBME in the single-relay system can be briefly summarized as follows [20].

Assume that the data are equally grouped into B blocks. Initially, the source broadcasts a codeword that corresponds to the first data block. Since the code rate is higher than the capacity of the link from the source to the destination, the destination is not able to recover the data reliably. Then the source and the relay cooperatively transmit more information about the first data block. In the meanwhile, the source "superimposes" a codeword that corresponds to the second data block. Finally, the destination is able to reliably recover the first data block from the two successive received blocks. After removing the effect of the first data block, the system returns to the initial state. This process iterates $B+1$ times until all B blocks of data are sent successfully.

We apply a similar strategy to the point-to-point communication system. We assume that the transmitter uses a short code. Initially, the transmitter sends a codeword that corresponds to the first data block. Since the short code is *weak*, the receiver is unable to recover reliably the data from the current received block. Hence the transmitter transmits the codeword (in its interleaved version) one more time. In the meanwhile, a fresh codeword that corresponds to the second data block is superimposed on the second block transmission. Finally, the receiver recovers the first data block from the two successive received blocks. This process iterates $B+1$ times until all B blocks of data are sent successfully. In practice, the receiver may use an iterative sliding-window decoding algorithm. The system performance can be analyzed in terms of the transmission memory and the input-output weight enumerating function (IOWEF) [5,6] of the BMST system, which can be computed from that of the short code using a trellis-based algorithm. Simulation results verify our analysis and show that remarkable coding gain can be obtained.

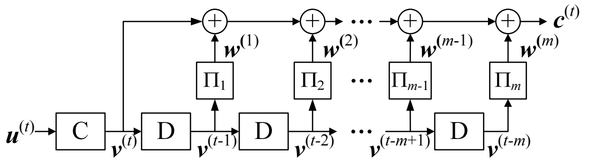

Fig. 1. Encoding structure of BMST with memory m.

II. Block Markov Superposition Transmission

A. Encoding Algorithm

We focus on binary linear codes. Let $\mathscr{C}[n,k]$ be a binary linear code with dimension k and length n, which is referred to as the *basic code* in this paper for convenience. Let $\boldsymbol{u}^{(0)}$, $\boldsymbol{u}^{(1)}$, $\cdots$, $\boldsymbol{u}^{(L-1)}$ be L blocks of data to be transmitted, where $\boldsymbol{u}^{(t)} \in \mathbb{F}_2^k$. The encoding algorithm with memory m is described as follows, see Fig. 1 for reference.

***Algorithm** 1:* Recursive Encoding of BMST

- *Initialization*: For $t < 0$, set $\boldsymbol{v}^{(t)} = \boldsymbol{0} \in \mathbb{F}_2^n$.
- *Recursion*: For $t = 0, 1, \cdots, L-1$,
  1) Encode $\boldsymbol{u}^{(t)}$ into $\boldsymbol{v}^{(t)} \in \mathbb{F}_2^n$ by the encoding algorithm of the basic code $\mathscr{C}$;
  2) For $1 \leq i \leq m$, interleave $\boldsymbol{v}^{(t-i)}$ by the i-th interleaver $\boldsymbol{\Pi}_i$ into $\boldsymbol{w}^{(i)}$;
  3) Compute $\boldsymbol{c}^{(t)} = \boldsymbol{v}^{(t)} + \sum_{1 \leq i \leq m} \boldsymbol{w}^{(i)}$, which is taken as the t-th block of transmission.
- *Termination*: For $t = L, L+1, \cdots, L+m-1$, set $\boldsymbol{u}^{(t)} = \boldsymbol{0} \in \mathbb{F}_2^k$ and compute $\boldsymbol{c}^{(t)}$ recursively (following Step. *Recursion*).

Remarks. The code rate is $\frac{kL}{n(L+m)}$, which is slightly less than that of the basic code $\mathscr{C}$. However, the rate loss is negligible for large L. It can be checked that the BMST system is a special class of convolutional codes [21] by replacing $\boldsymbol{G}_0$ and $\boldsymbol{G}_i$ in [21, (1.71)] with the generator matrix $\boldsymbol{G}$ of the basic code and $\boldsymbol{G}\boldsymbol{\Pi}_i (i = 1, \cdots, m)$, respectively. Unlike commonly accepted classical convolutional codes, the codes in this paper typically have large k and (hence) large constraint lengths.

B. Decoding Algorithm

For simplicity, we assume that $\boldsymbol{c}^{(t)}$ is modulated using binary-phase shift-keying (BPSK) and transmitted over an additive white Gaussian noise (AWGN) channel, resulting in a received vector $\boldsymbol{y}^{(t)}$. In more general settings, we assume that the *a posteriori* probabilities $\Pr\{c_j^{(t)} = 0, 1 | \boldsymbol{y}^{(t)}\}$ are computable[1], where $c_j^{(t)}$ is the j-th component of $\boldsymbol{c}^{(t)}$. Since the t-th data block $\boldsymbol{u}^{(t)}$ is involved in m consecutive transmissions $\boldsymbol{c}^{(t+i)}$, $0 \leq i \leq m$, we will use a sliding-window decoding algorithm. That is, the data block $\boldsymbol{u}^{(t)}$ will be decoded using the received blocks $\{\boldsymbol{y}^{(t+i)}, 0 \leq i \leq \min\{d, L+m-1-t\}\}$. Usually, we take the decoding delay $d \geq m$.

To describe the decoding algorithm, we represent the system by a normal graph [22]. Fig. 2 shows an example of a BMST system with $L = 4$ and $m = 2$. In the normal graph, nodes (represented by squares) represent system constraints

[1]The computation in this step is irrelevant to the code constraints but depends only on the modulation and the channel.

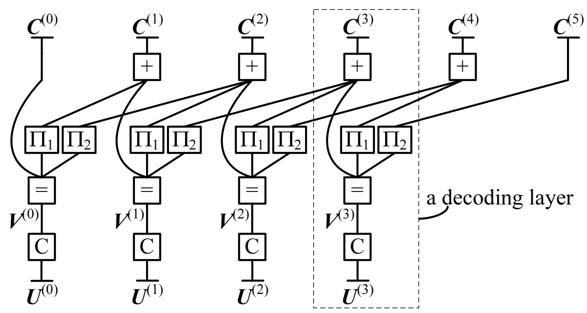

Fig. 2. The normal graph of a code with $L = 4$ and $m = 2$.

that must be fulfilled by all connected variables (represented by edges). A node serves as a message processor, which takes as input all messages collected from its connecting edges and delivers as output *extrinsic* messages to all its connecting edges (except some *half* edges). In our simulations, for a discrete random variable, we take its probability mass function (pmf) as the associated message. There are four types of nodes in the normal graph of the BMST system.

- *Node* $\boxed{\mathrm{C}}$: The node $\boxed{\mathrm{C}}$ represents the constraint that $\boldsymbol{v}^{(t)}$ must be a codeword of $\mathscr{C}$ that corresponds to $\boldsymbol{u}^{(t)}$. In practice, $\boldsymbol{u}^{(t)}$ is usually assumed to be independent and uniformly distributed over $\mathbb{F}_2^k$. Assume that the messages associated with $\boldsymbol{v}^{(t)}$ are available from the node $\boxed{=}$. The node $\boxed{\mathrm{C}}$ performs a soft-in-soft-out (SISO) decoding algorithm to compute the extrinsic messages. The extrinsic messages associated with $\boldsymbol{v}^{(t)}$ are fed back to the node $\boxed{=}$, while the extrinsic messages associated with $\boldsymbol{u}^{(t)}$ can be used to make decisions on the transmitted data.
- *Node* $\boxed{=}$: The node $\boxed{=}$ represents the constraint that all connecting variables must take the same realizations. The message processing/passing algorithm of the node $\boxed{=}$ is the same as that of the variable node in an LDPC code.
- *Node* $\boxed{\Pi_i}$: The node $\boxed{\Pi_i}$ represents the i-th interleaver, which interleaves or de-interleaves the input messages.
- *Node* $\boxed{+}$: The node $\boxed{+}$ represents the constraint that all connecting variables must be added up to zero over $\mathbb{F}_2$. The message processing/passing algorithm at $\boxed{+}$ is similar to that at the check node in an LDPC code. The only difference is that the messages associated with the half edge are available from the channel observations.

The iterative sliding-window algorithm with decoding delay $d \geq 0$ can be described as a message processing/passing algorithm over a subgraph containing $d+1$ *layers*, where each layer consists of a node of type $\boxed{\mathrm{C}}$, a node of type $\boxed{=}$, m nodes of type $\boxed{\Pi}$, and a node of type $\boxed{+}$, see Fig. 2 for reference. In our simulations, we use the following schedule to implement the decoding algorithm.

***Algorithm** 2:* Iterative Sliding-window Decoding of BMST

- *Initialization*: All messages over the intermediate edges are initialized as uniformly distributed variables. Assume that $\boldsymbol{y}^{(t)}, 0 \leq t \leq d-1$ have been received. Considering only the channel constraint, compute the *a posteriori* probabilities associated with $\boldsymbol{c}^{(t)}$ from the received vector

$\boldsymbol{y}^{(t)}$ for $0 \le t \le d-1$.

- *Sliding-window decoding*: For $t = 0, 1, \cdots, L-1$, the decoder takes as input the received vector $\boldsymbol{y}^{(t+d)}$, and delivers as output the estimated data block $\hat{\boldsymbol{u}}^{(t)}$. This can be done by first computing the *a posteriori* probabilities associated with $\boldsymbol{c}^{(t+d)}$ from the received vector $\boldsymbol{y}^{(t+d)}$ and then repeating the following steps at most $I_{\max}$ times.

Iteration: For $I = 1, 2, \cdots, I_{\max}$,

1) *Forward recursion*: For $i = 0, 1, \cdots, d$, the $(t+i)$-th layer performs a message processing/passing algorithm scheduled as

$$\boxed{+} \rightarrow \boxed{\Pi} \rightarrow \boxed{=} \rightarrow \boxed{\mathrm{C}} \rightarrow \boxed{=} \rightarrow \boxed{\Pi} \rightarrow \boxed{+}.$$

   In the above procedure, the message processor at each node takes as input all available messages from connecting edges and delivers as output extrinsic messages to connecting edges. Hence the messages from adjacent layers are utilized, and the messages to adjacent layers are updated by considering both the constraints in the $(t+i)$-th layer and the received vector $\boldsymbol{y}^{(t+i)}$.

2) *Backward recursion*: For $i = d, d-1, \cdots, 0$, the $(t+i)$-th layer perform a message processing/passing algorithm scheduled as

$$\boxed{+} \rightarrow \boxed{\Pi} \rightarrow \boxed{=} \rightarrow \boxed{\mathrm{C}} \rightarrow \boxed{=} \rightarrow \boxed{\Pi} \rightarrow \boxed{+}.$$

3) *Make decisions*: Make hard decisions on $\boldsymbol{v}^{(t)}$ and $\boldsymbol{u}^{(t)}$. If $\hat{\boldsymbol{v}}^{(t)}$ is a valid codeword in $\mathscr{C}$, report a successful decoding and exit the iteration.

Remark. If necessary, we may assume that the basic code is a concatenated code with a cyclic redundancy check (CRC) code as the outer code.

III. Coding Gain Analysis

To analyze the "extra" coding gain obtained by the BMST, we need to find bounds on the performance of the BMST system. Before doing this, we need to point out that the "extra" coding gain may be negative in the high BER region due to the possible error propagations. Let $p_b = f_o(\gamma_b)$ be the performance function under a given decoding algorithm of the basic code $\mathscr{C}$, where p_b is the BER and $\gamma_b \triangleq E_b/N_0$ in dB. Since $\mathscr{C}$ is short, we assume that $p_b = f_o(\gamma_b)$ is available. For example, if $\mathscr{C}$ is a terminated convolutional code, the performance function under the *maximum a posteriori probability* (MAP) decoding can be evaluated by performing the BCJR algorithm [23]. Let $p_b = f_n(\gamma_b)$ be the performance function corresponding to the BMST system.

A. Genie-Aided Lower Bound on BER

The performance of the BMST under MAP decoding is determined by $\Pr\{u_j^{(t)}|\boldsymbol{y}\}$. By Bayes' rule,

$$\Pr\{u_j^{(t)}|\boldsymbol{y}\} = \sum_{\boldsymbol{u}'} \Pr\{\boldsymbol{u}'|\boldsymbol{y}\}\Pr\{u_j^{(t)}|\boldsymbol{u}', \boldsymbol{y}\},$$

where the summation is over $\boldsymbol{u}' = \{\boldsymbol{u}^{(i)}, t-m \le i \le t+m, i \ne t\}$. Hence the BER performance can be lower-bounded by, taking into account the rate loss,

$$f_n(\gamma_b) \ge f_o(\gamma_b + 10\log_{10}(m+1) - 10\log_{10}(1+m/L)),$$

which holds because $\boldsymbol{v}^{(t)}$ is transmitted $m+1$ times provided that the transmitted data $\boldsymbol{u}'$ are known at the receiver. Furthermore, noticing that $\Pr\{\boldsymbol{u}'|\boldsymbol{y}\} \approx 1$ for the transmitted data block $\boldsymbol{u}'$ in the low error rate region, we can expect that

$$f_n(\gamma_b) \approx f_o(\gamma_b + 10\log_{10}(m+1) - 10\log_{10}(1+m/L))$$

as γ_b increases. That is, the maximum coding gain can be $10\log_{10}(m+1)$ dB for large L in the low error rate region.

B. Upper Bound on BER

To upper-bound the BER performance, we present a method to compute the IOWEF of the BMST. Let the IOWEF of the basic code $\mathscr{C}$ be given as

$$B(X,Y) \triangleq \sum_{i,j} B_{i,j} X^i Y^j, \tag{1}$$

where X, Y are two dummy variables and $B_{i,j}$ denotes the number of codewords having a Hamming weight j when the corresponding input information sequence having a Hamming weight i. Similarly, denote by $A(X,Y)$ the IOWEF of the BMST system. We have

$$\begin{aligned} A(X,Y) &= \sum_{\boldsymbol{u}} X^{W_H(\boldsymbol{u})} Y^{W_H(\boldsymbol{c})} \\ &= \sum_{\boldsymbol{u}} \prod_{t=0}^{L+m-1} X^{W_H(\boldsymbol{u}^{(t)})} Y^{W_H(\boldsymbol{c}^{(t)})}, \end{aligned}$$

where $W_H(\cdot)$ represents the Hamming weight and the summation is over all $\boldsymbol{u} = (\boldsymbol{u}^{(0)}, \cdots, \boldsymbol{u}^{(L-1)}, \boldsymbol{0}, \cdots, \boldsymbol{0})$. Since it is a sum of products, $A(X,Y)$ can be computed by a trellis-based algorithm over the polynomial ring. In the following, we take $m = 1$ as an example to describe the trellis. We further assume that the interleaver is chosen independently and uniformly at random for each transmission block $\boldsymbol{c}^{(t)}$. With this assumption, we can see that $W_H(\boldsymbol{c}^{(t)})$ is a random variable which is sensitive to neither $\boldsymbol{v}^{(t-1)}$ nor $\boldsymbol{v}^{(t)}$ but depends *only* on their Hamming weights $p = W_H(\boldsymbol{v}^{(t-1)})$ and $q = W_H(\boldsymbol{v}^{(t)})$. Specifically, we have

$$W_H(\boldsymbol{c}^{(t)}) = p + q - 2r \tag{2}$$

with probability

$$\Pr\{W_H(\boldsymbol{c}^{(t)}) = p+q-2r\} = \frac{\binom{p}{r}\binom{n-p}{q-r}}{\binom{n}{q}}, \tag{3}$$

where

$$r = \begin{cases} 0, 1, \cdots, \min(p,q), & p+q \le n \\ p+q-n, \cdots, \min(p,q), & p+q > n \end{cases}. \tag{4}$$

The trellis is time-invariant. At stage t, the trellis has $n+1$ states, each of which records the Hamming weight $W_H(\boldsymbol{v}^{(t-1)})$. Emitting from each state there are $n+1$ branches,

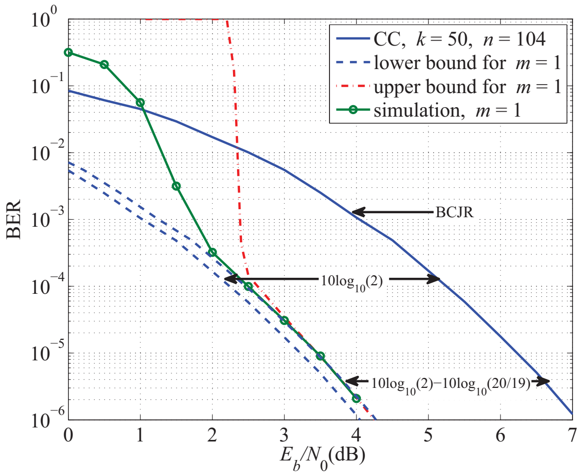

Fig. 3. Performance of the BMST system in Example 1.

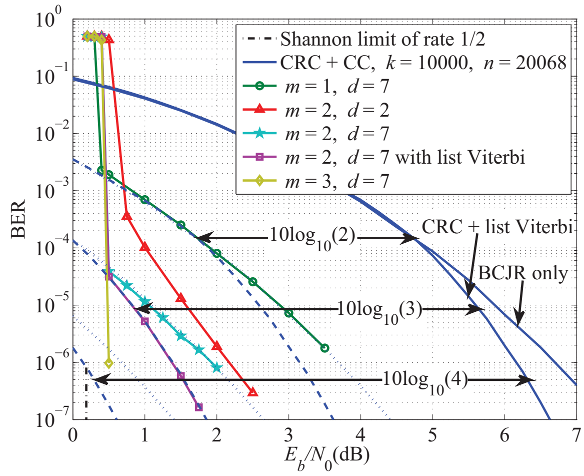

Fig. 4. Performance of the BMST system in Example 2.

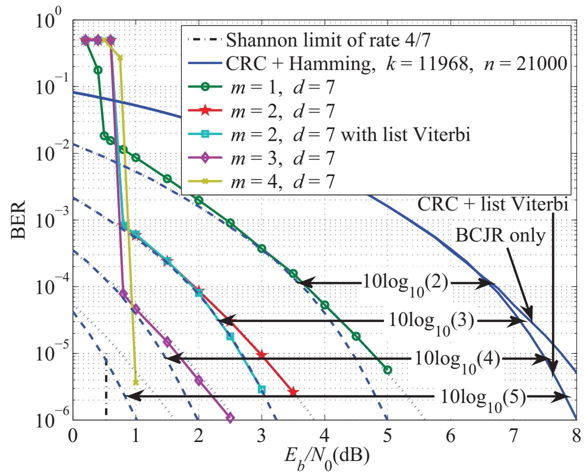

Fig. 5. Performance of the BMST system in Example 3.

each of which corresponds to the Hamming weight $W_H(\boldsymbol{v}^{(t)})$. To each branch $p \to q$, we assign a "metric"

$$\gamma_{p\to q} = \sum_r \Pr\{W_H(\boldsymbol{c}^{(t)}) = p+q-2r\} \sum_j B_{j,q} X^j Y^{p+q-2r}.$$

Then $A(X,Y)$ can be calculated recursively by performing a forward trellis-based algorithm [24] over the polynomial ring.

***Algorithm** 3:* Computing IOWEF of BMST with $m = 1$

1) Initialize $\alpha_0(p) = \sum_j B_{j,p} X^j Y^p$, $p \in \{0, 1, \cdots, n\}$.
2) For $t = 0, 1, \cdots, L-1$,

$$\alpha_{t+1}(q) = \sum_{p:p\to q} \alpha_t(p)\gamma_{p\to q},$$

where $q \in \{0, 1, \cdots, n\}$.

3) At time L, we have $A(X,Y) = \alpha_L(0)$.

Given $A(X,Y)$, the upper bound for the BER of the BMST system can be calculated by a similar improved union bound [25, (22)].

IV. Numerical Results

In this section, we present several construction examples, where all simulations are conducted by assuming BPSK modulation and AWGN channels. All the interleavers used in these examples are S-random [26] interleavers (randomly generated but fixed) with parameter $S = \lfloor\sqrt{(n/4)}\rfloor$, where $\lfloor x \rfloor$ stands for the maximum integer that is not greater than x. The BCJR algorithm is performed as the SISO decoding algorithm for the basic code (or its inner code).

Example 1: The basic code $\mathscr{C}$ is a terminated systematic encoded 4-state $(2,1,2)$ convolutional code (CC) with dimension $k = 50$ and length $n = 104$. We take $m = 1, L = 19$ for encoding and $d = 19$ (the whole frame), $I_{\max} = 18$ for decoding. Simulation results are shown in Fig. 3, which match well with the bounds in the low BER region.

Example 2: The basic code $\mathscr{C}$ is a concatenated code with $k = 10000$ and $n = 20068$, where the outer code is a 32-bit CRC code and the inner code is a terminated 4-state $(2,1,2)$ convolutional code. Simulation results for $L = 1000$ and $I_{\max} = 18$ are shown in Fig. 4. We can see that 1) given the encoding memory m, the performance can be improved by increasing the decoding delay d and 2) the performance in the low BER region can be improved by increasing m.

Example 3: The basic code $\mathscr{C}$ is a concatenated code with $k = 11968$ and $n = 21000$, where the outer code is a 32-bit CRC code and the inner code is the Cartesian product of Hamming code $[7,4]^{3000}$. Simulation results for $L = 1000$, $d = 7$ and $I_{\max} = 18$ are shown in Fig. 5. We can see that the BMST of the Hamming code has a similar behavior to the BMST of a convolutional code. With $m = 4$ and $d = 7$, an extra coding gain of 6.4 dB is obtained at BER 10^{-5}.

Remarks. From Fig. 4 and Fig. 5, we can see that the simulation results match well with the lower bounds derived from the BCJR-only curves but diverge from the bounds derived from the list-Viterbi curves. The reason is as follows. During the iterative sliding-window decoding of the BMST systems, the CRC code serves only as an error-detection code to exit the iteration at the right time and is less useful to enhance the error performance. To verify this, we have simulated an algorithm that performs list Viterbi [27] algorithm when the iterative sliding-window decoding is unsuccessful after $I_{\max}$

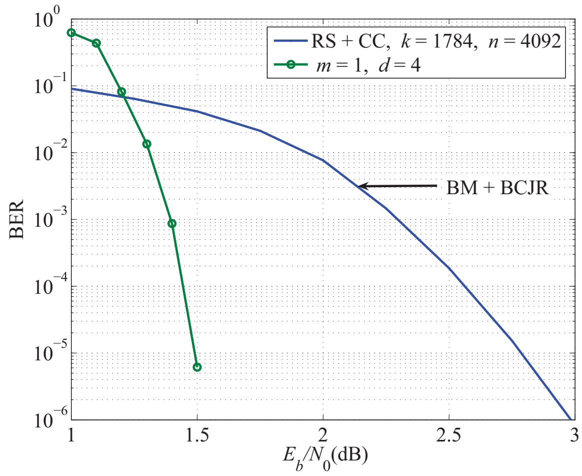

Fig. 6. Performance of the BMST system in Example 4.

iterations. As expected, the simulation results match well with the bounds derived from the list-Viterbi curves. As examples, see Fig. 4 and Fig. 5 with $m = 2$ and $d = 7$.

Example 4: In this example, we show that the BMST can also be applied to codes other than short codes. The basic code $\mathscr{C}$ is the Consultative Committee on Space Data System (CCSDS) standard code [28] with $k = 1784$ and $n = 4092$, where the outer code is a [255, 223] Reed-Solomon (RS) code over $\mathbb{F}_{256}$ and the inner code is a terminated 64-state $(2, 1, 6)$ convolutional code. The RS code not only removes the possible residual errors after the iterative sliding-window decoding of the inner code but also ensures (with high probability) the correctness of successfully decoded codewords. The simulation results with $L = 100$, $m = 1$, $d = 4$ and $I_{\max} = 18$ are shown in Fig. 6. The extra coding gain is about 1.3 dB at BER 10^{-5}.

V. Conclusion

In this paper, we presented a new method called block Markov superposition transmission (BMST) for constructing long codes from short codes. The encoding process can be as fast as the short code, while the decoding has a fixed delay. The coding gain of the BMST system is analyzed and verified by simulations. A nice property of the BMST is that its performance in the low BER region can be approximately predicted. A future work is to investigate the possible connections of the simple lower bound to the so-called threshold saturation phenomenon of the spatially coupled codes [29].

Acknowledgment

This work is supported by the 973 Program (No.2012CB316100) and the NSF (No.61172082) of China. The authors are grateful to Prof. Baoming Bai from Xidian University for useful discussions.

References

[1] F. J. MacWilliams and N. J. A. Sloane, *The theory of error-correcting codes*. Amsterdam, The Netherlands: North Holland, 1977.

[2] P. Elias, "Error-free coding," *IRE Trans. Inform. Theory*, vol. 4, pp. 29–37, Sept. 1954.

[3] G. D. Forney, Jr., *Concatenated codes*. Cambridge, MA: MIT Press, 1966.

[4] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo codes," in *Proc. of ICC'93*, vol. 2, Geneva, Switzerland, May 1993, pp. 1064–1070.

[5] S. Benedetto and G. Montorsi, "Unveiling turbo codes: Some results on parallel concatenated coding schemes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 409–428, Mar. 1996.

[6] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Serial concatenation of interleaved codes: Performance analysis, design, and iterative decoding," *IEEE Trans. Inform. Theory*, vol. 44, pp. 909–926, May 1998.

[7] D. Divsalar, H. Jin, and R. J. McEliece, "Coding theorems for 'turbo-like' codes," in *Proc. of the Annual Allerton Conference on Communication Control and Computing*, vol. 36, Sept. 1998, pp. 201–210.

[8] L. Ping, X. Huang, and N. Phamdo, "Zigzag codes and concatenated zigzag codes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 800–807, Feb. 2001.

[9] A. Abbasfar, D. Divsalar, and K. Yao, "Accumulate-repeat-accumulate codes," *IEEE Trans. Commun.*, vol. 55, pp. 692–702, Apr. 2007.

[10] S. Tong, L. Ping, and B. Bai, "Near Shannon limit precoded concatenated zigzag codes," in *International Symposium on Turbo Codes and Iterative Information Processing*, Aug. 2012, pp. 126–130.

[11] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA: MIT Press, 1963.

[12] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inform. Theory*, vol. 45, pp. 399–431, Mar. 1999.

[13] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 619–637, Feb. 2001.

[14] Y. Kou, S. Lin, and M. Fossorier, "Low-density parity-check codes based on finite geometries: A rediscovery and new results," *IEEE Trans. Inform. Theory*, vol. 47, pp. 2711 –2736, Nov. 2001.

[15] L. Lan, L. Zeng, Y. Tai, L. Chen, S. Lin, and K. Abdel-Ghaffar, "Construction of quasi-cyclic LDPC codes for AWGN and binary erasure channels: A finite field approach," *IEEE Trans. Inform. Theory*, vol. 53, pp. 2429–2458, Jul. 2007.

[16] A. J. Felström and K. S. Zigangirov, "Time-varying periodic convolutional codes with low-density parity-check matrix," *IEEE Trans. Inform. Theory*, vol. 45, pp. 2181–2191, Sept. 1999.

[17] A. E. Pusane, R. Smarandache, P. O. Vontobel, and D. J. Costello, "Deriving good LDPC convolutional codes from LDPC block codes," *IEEE Trans. Inform. Theory*, vol. 57, pp. 835–857, Feb. 2011.

[18] T. M. Cover and C. S. K. Leung, "An achievable rate region for the multiple-access channel with feedback," *IEEE Trans. Inform. Theory*, vol. 27, pp. 292–298, May 1981.

[19] T. M. Cover and A. A. E. Gamal, "Capacity theorems for the relay channel," *IEEE Trans. Inform. Theory*, vol. 25, pp. 572–584, Sept. 1979.

[20] F. M. J. Willems and E. C. V. D. Meulen, "The discrete memoryless multiple-access channel with cribbing encoders," *IEEE Trans. Inform. Theory*, vol. 31, pp. 313–327, May 1985.

[21] R. Johannesson and K. S. Zigangirov, *Fundamentals of convolutional coding*. Piscataway, NJ: IEEE Press, 1999.

[22] G. D. Forney Jr., "Codes on graphs: Normal realizations," *IEEE Trans. Inform. Theory*, vol. 47, 2001.

[23] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. 20, pp. 284–287, Mar. 1974.

[24] X. Ma and A. Kavčić, "Path partition and forward-only trellis algorithms," *IEEE Trans. Inform. Theory*, vol. 49, pp. 38–52, Jan. 2003.

[25] X. Ma, J. Liu, and B. Bai, "New techniques for upper-bounding the ML decoding performance of binary linear codes," *IEEE Trans. Commun.*, vol. 61, pp. 842–851, Mar. 2013.

[26] S. Dolinar and D. Divsalar, "Weight distributions for turbo codes using random and nonrandom permutations," *TDA Progress Report 42*, vol. 122, pp. 56–65, 1995.

[27] N. Seshadri and C.-E. W. Sundberg, "List Viterbi decoding algorithms with applications," *IEEE Trans. Commun.*, vol. 42, pp. 313–323, 1994.

[28] *TM Synchronization and Channel Coding*. The Consultative Committee for Space Data Systems (CCSDS) recommended standard, Aug. 2011.

[29] S. Kudekar, T. J. Richardson., and R. L. Urbanke, "Threshold saturation via spatial coupling: Why convolutional LDPC ensembles perform so well over the BEC," *IEEE Trans. Inform. Theory*, vol. 57, pp. 803–834, Feb. 2011.