On the Construction and MAP Decoding of Optimal Variable-Length Error-Correcting Codes

Ting-Yi Wu*, Po-Ning Chen*, Fady Alajaji† and Yunghsiang S. Han‡
*Dept. of Elec. Eng., Nat'l Chiao-Tung Univ., Taiwan (Email: mavericktywu@gmail.com, poning@faculty.nctu.edu.tw)
†Dept. of Mathematics and Statistics, Queen's Univ., Canada (Email: fady@mast.queensu.ca)
‡Dept. of Elec. Eng., Nat'l Taiwan Univ. of Science and Technology, Taiwan (Email: yshan@mail.ntust.edu.tw)

***Abstract*—In this paper, we present a novel algorithm that guarantees of finding a variable-length error-correcting code (VLEC) with minimal average codeword length for a fixed free distance d_{free}. We also propose a low complexity *maximum a posterior* (MAP) decoding algorithm for our codes under the premise that the receiver knows the number of codewords being transmitted. The resulting VLEC provides significant gains over other codes from the literature. When compared with separate source-channel tandem codes with identical d_{free}, such as a tandem code consisting of a Huffman source code concatenated with a $(2, 1, 4)$ tail-biting convolutional channel code, our system has only a 0.3 dB performance loss at a bit error rate of 10^{-5} while requiring significantly less decoding complexity.**

I. Introduction

In [1], Buttigieg explored properties of VLECs and confirmed that d_{free} affects the error performance of VLECs. Along with this finding, several approaches to construct VLECs targeting a given d_{free} were proposed in [5] and [7]. In [1], Buttigieg also modified the Viterbi algorithm (VA) to realize a MAP decoder for VLECs. Later in 2008, Huang *et al.* [3] proposed a trellis-based MAP priority-first search decoding algorithm for VLECs and empirically showed a complexity improvement over Buttigieg's MAP decoder. Recently, Savari and Kliewer [8] focused on minimizing the average codeword length of VLECs. In their design each codeword is required to have Hamming weight W, where W is a multiple of an integer ≥ 2, resulting in a class of VLECs with $d_{\text{free}} \geq 2$. In [9], an algorithm to develop VLECs with largest d_{free} was proposed under the premise that all codeword lengths are known in advance. A similar approach was used in [10] for constructing good error-correcting arithmetic codes. Other related work can be found in [12], [13].

In this work, a novel algorithm that can construct optimal VLECs in terms of minimal average codeword length under a fixed d_{free} is proposed. By assuming that the receiver knows the number of codewords that have been transmitted, an efficient two-phase MAP priority-first search decoder is presented for the constructed optimal codes. Our joint source-channel VLEC system outperforms the codes of [1] and [7]. We also compared it, in terms of performance and complexity, with a traditional tandem coding scheme that concatenates separately designed source and channel codes when transmitting a binary non-uniform memoryless source over a binary phase-shift keying (BPSK) modulated additive-white Gaussian noise (AWGN) channel.

This work was supported by the National Science Council of Taiwan under NSC 98-2221-E-009-060-MY3 and NSC 99-2221-E-009-076-MY3 and by the Natural Sciences and Engineering Research Council (NSERC) of Canada.

II. Preliminaries

Let $\mathcal{S} \triangleq \{\alpha_1, \alpha_2, \cdots, \alpha_K\}$ be the alphabet of a discrete memoryless source with respective source symbol probabilities p_1, p_2, $\cdots$ and p_K. Let $\mathcal{C} \triangleq \{c_1, c_2, \cdots, c_K\}$ be their corresponding binary VLEC codewords. The average codeword length of $\mathcal{C}$ is then given by $|\mathcal{C}| \triangleq \sum_{i=1}^{K} p_i |c_i|$, where $|c_i|$ is the length of codeword c_i.

A. Free Distance

Let $\mathcal{X}_{L,N} \triangleq \{x_1 x_2 \cdots x_L : \forall x_i \in \mathcal{C} \text{ and } \sum_{i=1}^{L} |x_i| = N\}$ be a set of bitstreams consisting of L codewords with overall length N. Define $\mathcal{X}_N \triangleq \bigcup_{i \geq 1} \mathcal{X}_{i,N}$. The d_{free} of $\mathcal{C}$ as defined in [1] is given by

$$d_{\text{free}}(\mathcal{C}) \triangleq \min\{d(a,b) : a, b \in \mathcal{X}_N \text{ for some } N \text{ and } a \neq b\},$$

where $d(a,b)$ denotes the Hamming distance between bitstreams a and b.

B. MAP Decoding Criterion

Assume the sequence of codewords of overall length N is transmitted over a binary-input memoryless channel and the received vector $\boldsymbol{r} \triangleq (r_1, r_2, \cdots, r_N)$. Define the hard decision of r_i as

$$y_i \triangleq \begin{cases} 1 & \text{if } \phi_i < 0 \\ 0 & \text{otherwise} \end{cases},$$

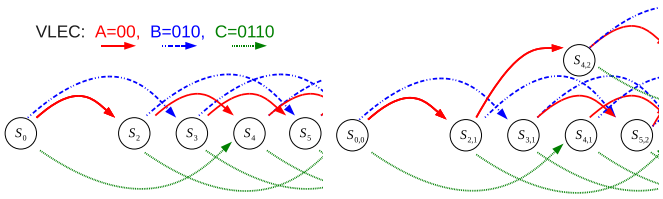
where $\phi_i \triangleq \ln[\Pr(r_i|0)/\Pr(r_i|1)]$. It can then be derived [3] that, with $\oplus$ denoting modulo-2 addition, the MAP decision $\hat{\boldsymbol{v}}$ satisfies

$$\sum_{i=1}^{N} (y_i \oplus \hat{v}_i)|\phi_i| - \ln \Pr(\hat{\boldsymbol{v}}) \leq \sum_{i=1}^{N} (y_i \oplus v_i)|\phi_i| - \ln \Pr(\boldsymbol{v})$$

for all $\boldsymbol{v} \in \begin{cases} \mathcal{X}_N, \text{if the receiver only knows } N; \\ \mathcal{X}_{L,N}, \text{if the receiver knows both } L \text{ and } N. \end{cases}$

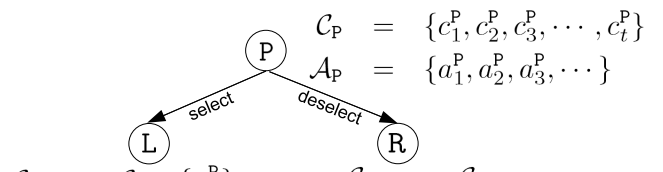
C. Trellis Diagram

In [1], Buttigieg employed a VLEC decoding trellis $\mathcal{T}_N$ as illustrated in Fig. 1(a) for $\mathcal{C} = \{00, 010, 0110\}$, in which state S_j denotes the number of bits decoded thus far is j. If the receiver knows both L and N, we can have an extension trellis $\mathcal{T}_{L,N}$, where $S_{i,j}$ denotes the numbers of decoded codewords and decoded bits are i and j, respectively, as shown in Fig. 1(b).



(a) Trellis $\mathcal{T}_N$ (b) Trellis $\mathcal{T}_{L,N}$

Fig. 1. Trellis representations of a VLEC. The red-color, blue-color and green-color arrows correspond respectively to the transition of transmitting codewords A, B and C.



$$\mathcal{C}_{\mathtt{P}} = \{c_1^{\mathtt{P}}, c_2^{\mathtt{P}}, c_3^{\mathtt{P}}, \cdots, c_t^{\mathtt{P}}\}$$
$$\mathcal{A}_{\mathtt{P}} = \{a_1^{\mathtt{P}}, a_2^{\mathtt{P}}, a_3^{\mathtt{P}}, \cdots\}$$
$$\mathcal{C}_{\mathtt{L}} = \mathcal{C}_{\mathtt{P}} \cup \{a_1^{\mathtt{P}}\} \qquad \mathcal{C}_{\mathtt{R}} = \mathcal{C}_{\mathtt{P}}$$
$$\mathcal{A}_{\mathtt{L}} = \{a_1^{\mathtt{L}}, a_2^{\mathtt{L}}, a_3^{\mathtt{L}}, \cdots\} \qquad \mathcal{A}_{\mathtt{R}} = \mathcal{A}_{\mathtt{P}} \setminus \{a_1^{\mathtt{P}}\}$$

Fig. 2. Relation between a parent node and its children in a search tree.

III. Code Construction

Recently, Huang *et al.* [4] proposed a new search algorithm to construct optimal reversible variable-lengths codes. We herein observe that by modifying the algorithm in [4], an algorithm that guarantees finding optimal VLECs for a given d_{free}^* can be obtained. To demonstrate our algorithm, we first construct the search tree for VLECs, then describe the search algorithm step by step.

A. Search Tree

To construct a VLEC with K elements, a search tree is used in which each node $\mathtt{X}$ contains three elements denoted by $\{\mathcal{C}_{\mathtt{X}}, \mathcal{A}_{\mathtt{X}}, f(\mathtt{X})\}$. Here $\mathcal{C}_{\mathtt{X}} = \{c_1^{\mathtt{X}}, c_2^{\mathtt{X}}, \cdots, c_t^{\mathtt{X}}\}$ denotes the set of t codewords that have been selected for the desired VLEC. $\mathcal{A}_{\mathtt{X}} = \{a_1^{\mathtt{X}}, a_2^{\mathtt{X}}, \cdots\}$ is the set of possible bitstreams, excluding all bitstreams whose prefix is in $\mathcal{C}_{\mathtt{X}}$, which could be a codeword for $\mathcal{C}_{\mathtt{X}}$ (candidate codewords) listed in order of non-decreasing lengths: $|a_1^{\mathtt{X}}| \le |a_2^{\mathtt{X}}| \le \cdots$. Finally, $f(\mathtt{X})$ is used as a metric for the search algorithm; it is defined as

$$f(\mathtt{X}) = \sum_{i=1}^{t} p_i \cdot |c_i^{\mathtt{X}}| + \sum_{i=t+1}^{K} p_i \cdot |a_{i-t}^{\mathtt{X}}|.$$

The search tree is constructed as a binary tree, and the relation between a parent node and its children is illustrated in Fig. 2. Specifically, for a parent node $\mathtt{P}$, its left child $\mathtt{L}$ is obtained by adding the next candidate codeword $a_1^{\mathtt{P}}$ to $\mathcal{C}_{\mathtt{L}}$. Since $a_1^{\mathtt{P}}$ is now a codeword in $\mathcal{C}_{\mathtt{L}}$, the set $\mathcal{A}_{\mathtt{L}}$ needs to be updated by removing all bitstreams in $\mathcal{A}_{\mathtt{P}}$ whose prefix is $a_1^{\mathtt{P}}$. Hence, the left child $\mathtt{L}$ becomes

$$\begin{aligned}
\mathcal{C}_{\mathtt{L}} &= \mathcal{C}_{\mathtt{P}} \cup \{a_1^{\mathtt{P}}\} \\
\mathcal{A}_{\mathtt{L}} &= \{a_1^{\mathtt{L}}, a_2^{\mathtt{L}}, \cdots\} \\
&= \{a : a \in \mathcal{A}_{\mathtt{P}} \text{ and } a_1^{\mathtt{P}} \text{ is not prefix of } a\} \\
f(\mathtt{L}) &= \sum_{i=1}^{t} p_i \cdot |c_i^{\mathtt{P}}| + p_{t+1} \cdot |a_1^{\mathtt{P}}| + \sum_{i=t+2}^{K} p_i \cdot |a_{i-t-1}^{\mathtt{L}}|.
\end{aligned}$$

On the other hand, the right child $\mathtt{R}$ is obtained by rejecting the next candidate codeword $a_1^{\mathtt{P}}$ from its parent node. So, the right child $\mathtt{R}$ becomes

$$\begin{aligned}
\mathcal{C}_{\mathtt{R}} &= \mathcal{C}_{\mathtt{P}} \\
\mathcal{A}_{\mathtt{R}} &= \{a_2^{\mathtt{P}}, a_3^{\mathtt{P}}, \cdots\} = \mathcal{A}_{\mathtt{P}} \setminus \{a_1^{\mathtt{P}}\} \\
f(\mathtt{R}) &= \sum_{i=1}^{t} p_i \cdot |c_i^{\mathtt{P}}| + \sum_{i=t+1}^{K} p_i \cdot |a_{i-t+1}^{\mathtt{P}}|.
\end{aligned}$$

By traversing the path from the root node, every possible VLEC can be found in a leaf of the tree.

B. Finding an Optimal VLEC with a Fixed d_{free}^*

Since the search tree can be well-constructed, the priority-first search algorithm can be easily applied on it. To reduce the search space, the average codeword length of any existing VLEC with free distance no less than the d_{free}^* can be used as an upper bound, denoted by B, during the search process. The algorithm for finding an optimal VLEC is described as follows.

Step 1. Initialize the root node as

$$\begin{aligned}
\mathcal{C}_{\mathtt{root}} &= \emptyset \\
\mathcal{A}_{\mathtt{root}} &= \{0, 1, 00, 01, 10, 11, 000, 001, \cdots\} \\
f(\mathtt{root}) &= \sum_{i=1}^{K} p_i \cdot |a_i^{\mathtt{root}}|,
\end{aligned}$$

and push it into the Stack. Set upper bound B as the smallest average codeword length of existing VLECs with free distance no less than d_{free}^*.

Step 2. If the top node of the Stack has selected K codewords (i.e., $|\mathcal{C}_{\mathtt{top}}| = K$) and $d_{free}(\mathcal{C}_{\mathtt{top}}) = d_{free}^*$, then output $\mathcal{C}_{\mathtt{top}}$ as the optimal VLEC and the algorithm stops.

Step 3. Generate the two children of the top node as in Fig. 2 and then delete the top node from the Stack. If the left child has selected K codewords with its free distance $\ge d_{free}^*$ and its associated metric f is smaller than B, then update $B = f$.

Step 4. Discard the child node which satisfies any one of the following conditions:

1) *It has selected more than K codewords for $\mathcal{C}$;*
2) *There is no more candidate in $\mathcal{A}$ (i.e. $\mathcal{A} = \emptyset$);*
3) *The metric f is larger than B.*

Step 5. Insert the remaining children into the Stack, and reorder the Stack in order of ascending metrics. Go to Step 2.

Observation: It should be pointed out that the above code construction algorithm focuses only on VLECs that satisfy the prefix-free condition, i.e., VLECs for which no codeword can be a prefix of any other codeword. Although there exist non-prefix-free but uniquely decodable VLECs, their implementation may require extra mechanisms such as buffers; hence, they are generally regarded as less cost-effective. For this reason, we restrict ourselves to the search of optimal prefix-free VLECs as in most previous works [1], [5], [7]-[10].

C. Proof of Optimality

To show that the proposed algorithm can always find a VLEC with minimal average codeword length and free distance d^*_{free}, the following lemma is needed.

Lemma 1: The metric f of each node is not greater than its children:

$$f(\mathtt{P}) \le f(\mathtt{L}) \text{ and } f(\mathtt{P}) \le f(\mathtt{R}),$$

where node P is the parent of L, and R as shown in Fig. 2.

Proof: The candidate codewords of each node are listed in order of non-decreasing lengths (i.e., $\mathcal{A} = \{a_1, a_2, a_3, \cdots\}$ with $|a_1| \le |a_2| \le |a_3| \cdots$). For the left child L, $\mathcal{A}_{\mathtt{L}}$ is a subset of $\mathcal{A}_{\mathtt{P}} \setminus \{a_1^{\mathtt{P}}\}$. Hence $|a_{i+1}^{\mathtt{P}}| \le |a_i^{\mathtt{L}}|$ for all integers $i \ge 1$. Therefore,

$$\begin{aligned} f(\mathtt{P}) &= \sum_{i=1}^{t} p_i \cdot |c_i^{\mathtt{P}}| + \sum_{i=t+1}^{K} p_i \cdot |a_{i-t}^{\mathtt{P}}| \\ &= \sum_{i=1}^{t} p_i \cdot |c_i^{\mathtt{P}}| + p_{t+1} \cdot |a_1^{\mathtt{P}}| + \sum_{i=t+2}^{K} p_i \cdot |a_{i-t}^{\mathtt{P}}| \\ &\le \sum_{i=1}^{t} p_i \cdot |c_i^{\mathtt{P}}| + p_{t+1} \cdot |a_1^{\mathtt{P}}| + \sum_{i=t+2}^{K} p_i \cdot |a_{i-t-1}^{\mathtt{L}}| \\ &= f(\mathtt{L}). \end{aligned}$$

Since $|a_i^{\mathtt{P}}| \le |a_{i+1}^{\mathtt{P}}|$ for $i \ge 1$, for the right child R, we have

$$\begin{aligned} f(\mathtt{P}) &= \sum_{i=1}^{t} p_i \cdot |c_i^{\mathtt{P}}| + \sum_{i=t+1}^{K} p_i \cdot |a_{i-t}^{\mathtt{P}}| \\ &\le \sum_{i=1}^{t} p_i \cdot |c_i^{\mathtt{P}}| + \sum_{i=t+1}^{K} p_i \cdot |a_{i-t+1}^{\mathtt{P}}| \\ &= f(\mathtt{R}). \end{aligned}$$

■

The proposed algorithm repeatedly pops out the node with smallest f from the Stack. Suppose that the algorithm encounters the first top node which has selected K codewords and its free distance equals d^*_{free}; then by Lemma 1, no matter how the algorithm continues, extending any node in the Stack will generate a node with metric f no smaller then the top node. Hence, the algorithm yields an optimal VLEC.

IV. MAP Decoder

Usually, an MAP decoder for VLECs operates under the assumption that the receiver knows only the number of transmitted bits. However, if the receiver also knows the number of transmitted codewords, the error performance of the MAP decoder can be improved. As a result, the MAP decoder can operate on an extended trellis $\mathcal{T}_{L,N}$ as shown in Fig. 1(b). This extended trellis has much more nodes than the traditional trellis $\mathcal{T}_N$.

We then proceed by representing a path from $S_{0,0}$ to $S_{i,j}$ in trellis $\mathcal{T}_{L,N}$ by the codeword it traverses $\boldsymbol{x}_{(0,0)}^{(i,j)} \triangleq x_1 x_2 \cdots x_i \in \mathcal{X}_{i,j}$, where each $x_i \in \mathcal{C}$. We can also represent $\boldsymbol{x}_{(0,0)}^{(i,j)}$ by its equivalent binary stream $b_1 b_2 \cdots b_j$ and define the path metric of $\boldsymbol{x}_{(0,0)}^{(i,j)}$ as

$$g\left(\boldsymbol{x}_{(0,0)}^{(i,j)}\right) = \sum_{\ell=1}^{j} (y_\ell \oplus b_\ell)|\phi_\ell| - \ln \Pr\left(\boldsymbol{x}_{(0,0)}^{(i,j)}\right). \quad (1)$$

Then, the MAP decision with known L and N at the receiver can be found by applying the VA over trellis $\mathcal{T}_{L,N}$ with the path metric just defined. However, as the number of nodes in $\mathcal{T}_{L,N}$ grows dramatically even with moderate L and N, the decoding complexity of such a straightforward design is apparently infeasible.

A. Two-Phase MAP decoder

Inspired by the decoder algorithm proposed in [6], an alternative two-phase decoding design is therefore proposed in this work to search for the MAP decision over trellis $\mathcal{T}_{L,N}$. The proposed algorithm first applies the VA in a backward fashion on trellis $\mathcal{T}_N$, which has a considerably smaller number of nodes than $\mathcal{T}_{L,N}$, and retains the metric $h(S_j)$ of each reverse path ending at node S_j. If at the end of the first phase, the reverse path with the minimum metric contains L codewords, then this is the final decision, and there is no need to proceed to the second phase; otherwise, the second phase is performed. The first phase is described as follows.

Step 1. Associate a zero path metric to node S_N in $\mathcal{T}_N$, $h(S_N) = 0$.

Step 2. Apply the backward VA with path metric given by (1) from S_N on $\mathcal{T}_N$, and record the metric and survivor path for each state as $h(S_i)$ and $p_b(S_i)$, respectively.

Step 3. If the number of codewords correspond to $p_b(S_0)$ is equal to L, then output $p_b(S_0)$ as the MAP decision and the algorithm stops; otherwise, go to phase 2.

In the second phase, the decoding metric for path $\boldsymbol{x}_{(0,0)}^{(i,j)}$ in $\mathcal{T}_{L,N}$ is re-defined as

$$m\left(\boldsymbol{x}_{(0,0)}^{(i,j)}\right) = g\left(\boldsymbol{x}_{(0,0)}^{(i,j)}\right) + h\left(S_{i,j}\right),$$

where $h(S_{i,j}) = h(S_j)$. We can then apply the algorithm of [2] in a forward way on trellis $\mathcal{T}_{L,N}$ and determine the MAP decision. The second phase of the decoder is next described.

Step 1. Initialize the path metric of $\boldsymbol{x}_{(0,0)}^{(0,0)}$ as $m(\boldsymbol{x}_{(0,0)}^{(0,0)}) = h(S_0)$, and load it into the Open Stack.

Step 2. If the top node of the Open Stack reaches the final state $S_{L,N}$ in $\mathcal{T}_{L,N}$, then output its associated path as the MAP decision and the algorithm stops.

Step 3. Record the state of the top node in the Close Table, then extend the top node to all its successors and compute their metrics. Delete the top node from the Open Stack.

Step 4. Discard the successors that have been recorded in the Close Table. Discard the successors for which the number of decoded symbols exceeds L or the number of decoded bits exceeds N.

Step 5. Insert the remaining successors into the Open Stack and reorder the Open Stack in order of ascending metrics. Go to Step 2.

In the second phase described above, the *Open Stack*[1] is a data structure which stores all visited nodes and can easily access the node with minimum decoding metric. Unlike the *Open Stack*, the *Close Table* is used to record nodes that have been extended.

B. Proof of Optimality

The proof is similar to the one given in Section III-C, except that we need to prove that the path metric is non-decreasing along any path on trellis $\mathcal{T}_{L,N}$.

Lemma 2: In the second phase, the decoding metric is non-decreasing along any path on trellis $\mathcal{T}_{L,N}$, i.e.,

$$m\left(\boldsymbol{x}_{(0,0)}^{(i,j)}\right) \leq m\left(\boldsymbol{x}_{(0,0)}^{(i+1,j+m)}\right),$$

if there exists a codeword $c \in \mathcal{C}$ and $|c| = m$.

Proof: Based on the backward VA of the first phase, $h(S_j)$ is the minimal metric among all paths from level j to the final node; i.e.,

$$h(S_j) = \min_{i:\sum_{k=i+1}^{L}|x_k|=N-j \text{ with each } x_k \in \mathcal{C}} g\left(\boldsymbol{x}_{(i,j)}^{(L,N)}\right).$$

When there is a codeword $c \in \mathcal{C}$ and $|c| = m$, then

$$h(S_j) \leq g\left(\boldsymbol{x}_{(i,j)}^{(i+1,j+m)}\right) + h\left(S_{j+m}\right).$$

Therefore,

$$\begin{aligned} m\left(\boldsymbol{x}_{(0,0)}^{(i,j)}\right) &= g\left(\boldsymbol{x}_{(0,0)}^{(i,j)}\right) + h(S_j) \\ &\leq g\left(\boldsymbol{x}_{(0,0)}^{(i,j)}\right) + g\left(\boldsymbol{x}_{(i,j)}^{(i+1,j+m)}\right) + h\left(S_{j+m}\right) \\ &= g\left(\boldsymbol{x}_{(0,0)}^{(i+1,j+m)}\right) + h(S_{j+m}) \\ &= m\left(\boldsymbol{x}_{(0,0)}^{(i+1,j+m)}\right). \end{aligned}$$

■

V. Simulation Results over the AWGN Channel

In all simulations, the alphabet set $\mathcal{S}$ is obtained by grouping a block of 3 information bits generated from a binary non-uniform memoryless source with bit probabilities p_0 and $p_1 = 1 - p_0$. In Table I, we compare the VLECs found by the proposed method with Buttigieg's code [1] and the code by Wang *et al.* [7] for different values of p_0 and d_{free}^*. Since our proposed algorithm guarantees to find VLECs with minimal average codeword length under a fixed d_{free}^*, the resulting VLECs have a shorter average codeword length than any other code with identical free distance. Note that the algorithms of Buttigieg and Wang *et al.* generate identical VLEC when free distance equal to 5 and 7.

We next investigate the error performance[2] of different VLECs and coding schemes for a BPSK-modulated AWGN channel with average energy per information bit E_b and noise variance $N_0/2$. In all simulations, at least 100 block errors were counted to ensure the accuracy of the results.

TABLE I
Average codeword length per information bit of a 8-ary alphabet generated from a binary non-uniform (p_0) memoryless source

Algorithm	Buttigieg's [1]		Wang's [7]		Proposed	
p_0	0.7	0.8	0.7	0.8	0.7	0.8
$d_{\text{free}}^* = 3$	1.66	1.39	1.52	1.36	1.49	1.33
$d_{\text{free}}^* = 5$	2.15	1.97	2.15	1.97	2.11	1.86
$d_{\text{free}}^* = 7$	2.78	2.62	2.78	2.62	2.67	2.41

In Fig. 3, 30 information bits (i.e., 10 grouped symbols) are encoded by the VLECs with $d_{\text{free}}^* = 7$ of Table I. The dotted lines show the performance of the MAP decoder under the assumption that the receiver only knows the number of transmitted bits, N. The solid line portrays the MAP decoder's performance under the assumption that receiver knows both number of symbols, L, and transmitted bits, N. We clearly observe that VLECs found by the proposed method outperform the other VLECs; furthermore, about 0.5 dB in coding gain is realized by knowing L (in addition to N).

Fig. 4 presents the performance of VLECs found by the proposed method with $d_{\text{free}}^* = 7$ for different values of p_0 and L. The figure indicates that the VLECs are better when the source distribution is more biased and the block length is shorter.

In Fig. 5, we compare the VLECs found by the proposed method with $d_{\text{free}}^* = 7$ with a traditional tandem (separate) source-channel coding scheme. The information bits are generated using $p_0 = 0.8$, and ten 3-bit symbols ($L = 10$) are encoded and transmitted. The tandem system is a concatenation of Huffman code and a $(2, 1, 4)$ tail-biting convolutional code (TBCC) with generator polynomial $[27, 31]$ (in octal) and $d_{\text{free}}^* = 7$. The tandem system performs only about 0.3 dB better than the VLEC code at a BER of 10^{-5}. For signal-to-noise ratios (SNRs) $E_b/N_0 \leq 3$ dB, the VLEC has a better BER performance than the tandem system.

Finally, Table II compares the decoding complexity of different schemes for $p_0 = 0.8$ and $L = 10$. From the table, we remark that the two-phase MAP decoder has similar decoding complexity as VA-MAP on $\mathcal{T}_N$ while achieving about 0.5 dB coding gain in error performance. For identical error performance, the two-phase decoding algorithm spends almost 4 times less in branch computations than VA-MAP on $\mathcal{T}_{L,N}$. In comparison with the tandem scheme mentioned above, the two-phase MAP decoder requires less decoding complexity than the decoding of the $(2, 1, 4)$ TBCC component code by a PFSA decoder [6].

VI. Conclusion

In this work, an optimal search algorithm for prefix-free VLECs and an efficient MAP decoder are proposed. The resulting (optimal) VLECs outperform all existing VLECs with identical free distance and are comparable to a tandem scheme that concatenates a Huffman code with a tail-biting convolutional code. One advantage of the VLEC approach (as opposed to tandem coding) is that only one encoder and

[1]In our simulations, the *Open Stack* is implemented via the date structure *HEAP* [11]. One important property of *HEAP* is that it can access the node with minimal metric within a $O(\log(n))$ complexity, where n denotes the *HEAP*'s number of nodes.

[2]The error performance in terms of the information bit error rate (BER) is measured via the Levenshtein distance.

TABLE II
AVERAGE (AVG) AND MAXIMUM (MAX) NUMBERS OF BRANCH COMPUTATIONS

E_b/N_0		1 dB		2 dB		3 dB		4 dB		5 dB	
scheme		avg	max	avg	max	avg	max	avg	max	avg	max
code	decoder										
our VLEC with $d^*_{\text{free}} = 7$	VA-MAP on $\mathcal{T}_N$	459	768	459	768	459	768	459	768	459	768
	VA-MAP on $\mathcal{T}_{L,N}$	1651	2600	1651	2600	1651	2600	1651	2600	1651	2600
	2-phase MAP on $\mathcal{T}_{L,N}$	461	3047	460	1780	459	956	459	768	459	768
$(2, 1, 4)$ TBCC	PFSA in [6]	730	2189	704	1668	700	1408	699	1408	699	1408

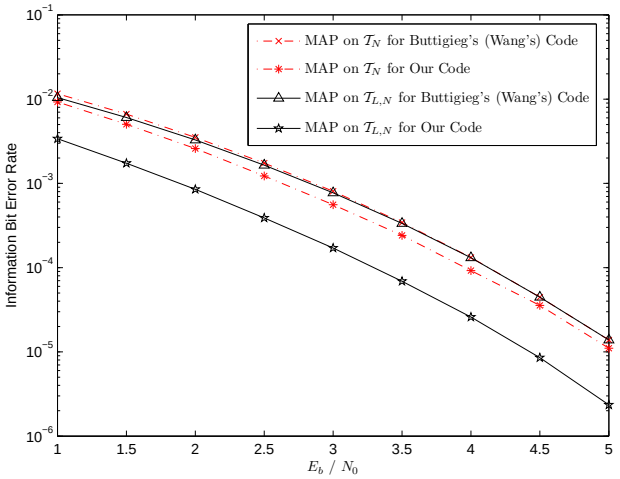


Fig. 3. BER for MAP decoding VLECs with $d^*_{\text{free}} = 7$. The block length $L = 10$ and the number of information bits is 30.

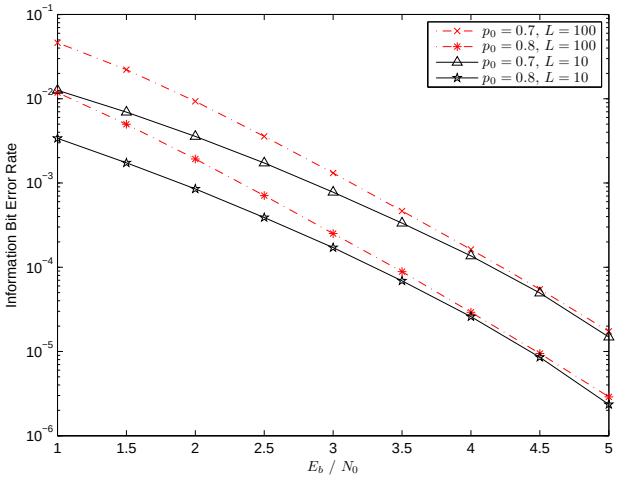


Fig. 4. BER for MAP decoding on $\mathcal{T}_{L,N}$ the VLECs found by the proposed method with $d^*_{\text{free}} = 7$ for different values of p_0 and block length L.

decoder are needed. It is known that the VLECs' free distance only affects their BER performance for high SNRs. Optimizing VLECs in terms of other parameters such as the number of codewords of weight equal to the free distance to further improve BER performance in the medium-to-low SNR range is an interesting future work.

REFERENCES

[1] V. Buttigieg, *Variable-Length Error-Correcting Codes*, Ph.D. thesis, Univ. of Manchester, England, 1995.

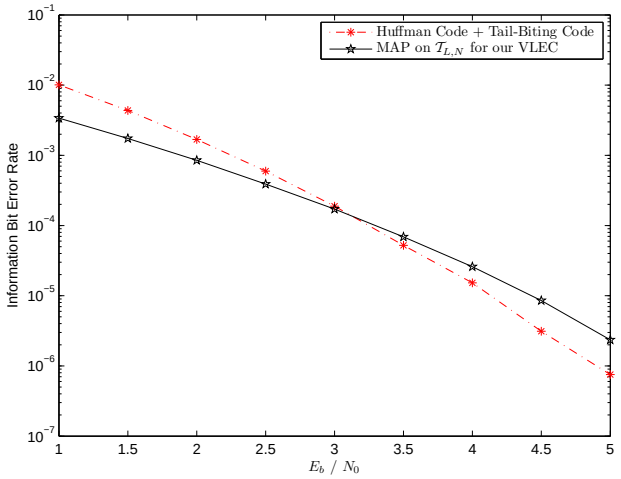


Fig. 5. BER of the VLECs found by the proposed method with $d^*_{\text{free}} = 7$ on $\mathcal{T}_{L,N}$ and the traditional tandem scheme (generator polynomial of tail-biting code is $[27, 31]$ in octal).

[2] Y. S. Han, P.-N. Chen and H.-B. Wu, "A maximum-likelihood soft-decision sequential decoding algorithm for binary convolutional codes," *IEEE Trans. Commun.*, vol. 50, no. 2, pp. 173-178, Feb. 2002.

[3] Y.-M. Huang, Y. S. Han and T.-Y. Wu, "Soft-decision priority-first decoding algorithms for variable-length error-correcting codes," *IEEE Comm. Letters*, vol. 12, no. 8, pp. 572-574, Aug. 2008.

[4] Y.-M. Huang, T.-Y. Wu and Y. S. Han, "An A*-based algorithm for constructing reversible variable-length codes with minimum average codeword length," *IEEE Trans. Commun.*, vol. 58, no. 11, pp. 3175-3185, Nov. 2010.

[5] C. Lamy and J. Paccaut, "Optimized constructions for variable-length error-correcting codes," *in IEEE Inform. Theory Workshop*, pp. 183-186, Nov. 2003.

[6] H.-T. Pai, Y. S. Han, T.-Y. Wu, P.-N. Chen and S.-L. Shieh, "Low-complexity ML decoding for convolutional tail-biting codes," *IEEE Comm. Letters*, vol. 12, no. 12, pp. 883-885, Dec. 2008.

[7] J. Wang, L.-L. Yang and L. Hanzo, "Iterative construction of reversible variable-length codes and variable-length error-correcting codes," *IEEE Comm. Letters*, vol. 8, no. 11, pp. 671-673, Nov. 2004.

[8] S. A. Savari and J. Kliewer, "When Huffman meets Hamming: A class of optimal variable-length error correcting codes," *in Proc. Data Compression Conf.*, pp. 327-336, Snowbird, Utah, Mar. 2010.

[9] A. Diallo, C. Weidmann and M. Kieffer, "Optimizing the Free Distance of Error-Correcting Variable-Length Codes," *in Proc. IEEE Int. Workshop Multimedia Signal Proc. (MMSP)*, St. Malo, France, Oct. 2010.

[10] A. Diallo, C. Weidmann and M. Kieffer, "Efficient computation and optimization of the free distance of variable-length finite-state joint source-channel codes," to appear in *IEEE Trans. Commun.*, Apr. 2011.

[11] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT press, Cambridge, MA 2001.

[12] R. Bauer and J. Hagenauer, "Iterative source/channel-decoding using reversible variable length codes," *in Proc. Data Compression Conf.*, pp. 93-102, Snowbird, Utah, Mar. 2000.

[13] V. B. Balakirsky, "Joint source-channel coding with variable length codes," *in Proc. IEEE Int. Symposium on Information Theory*, p. 419, Ulm, Germany, July 1997.