On the Design of Variable-Length Error-Correcting Codes

Ting-Yi Wu, *Student Member, IEEE,* Po-Ning Chen, *Senior Member, IEEE,*
Fady Alajaji, *Senior Member, IEEE,* and Yunghsiang S. Han, *Fellow, IEEE*

***Abstract*—A joint source-channel coding problem that combines the efficient compression of discrete memoryless sources with their reliable communication over memoryless channels via binary prefix-free variable-length error-correcting codes (VLECs) is considered. Under a fixed free distance constraint, a priority-first search algorithm is devised for finding an optimal VLEC with minimal average codeword length. Two variations of the priority-first-search-based code construction algorithm are also provided. The first one improves the resilience of the developed codes against channel noise by additionally considering a performance parameter $B_{d_{\text{free}}}$ without sacrificing optimality in average codeword length. In the second variation, to accommodate a large free distance constraint as well as a large source alphabet such as the 26-symbol English data source, the VLEC construction algorithm is modified with the objective of significantly reducing its search complexity while still yielding near-optimal codes. A low-complexity sequence *maximum a posteriori* (MAP) decoder for all VLECs (including our constructed optimal code) is then proposed under the premise that the receiver knows the number of codewords being transmitted. Simulations show that the realized optimal and suboptimal VLECs compare favorably with existing codes in the literature in terms of coding efficiency, search complexity and error rate performance.**

***Index Terms*—Joint source-channel coding, variable-length codes, error resilient data compression, sequence maximum a posteriori decoding, average codeword length, symbol error rate.**

I. Introduction

ONE of Shannon's key contributions in information theory is the separation principle for source-channel coding [27], which states that the source and channel coding operations can be separately designed and performed in tandem without affecting the system's optimality for reliably transmitting a data source over a noisy channel. However, this result hinges on the assumption that unlimited complexity and coding delay can be afforded by the system, which is unrealistic in today's resource constrained communication systems. It is indeed well-known via both analytical and empirical studies (e.g., see [1], [2], [14], [33] and the references therein) that joint source-channel coding (JSCC) can significantly outperform separate source-channel coding (SSCC), particularly when the system has stringent delay and complexity restrictions. JSCC, which may use codes of fixed or variable length, is typically realized in two ways: by coordinating the source and channel coding functions in tandem or by combining them within a single step (examples of various JSCC schemes can be found in [33]). In this paper, we focus on variable-length single-step JSCC with the objective of designing optimal or close-to-optimal variable-length error-correcting codes (VLEC) with low complexity for the efficient compression and communication of data sources in the presence of channel noise. Here optimality is interpreted as achieving minimal average codeword length among all VLEC designs subject to a fixed free-distance constraint. The successful development of such VLECs, which play the dual role of good data compression and error-correcting codes, provides an interesting alternative to the classical SSCC scheme, particularly when the system's complexity can be significantly reduced without degrading its error performance.

First introduced in [17], [5], [6], VLECs were thoroughly investigated by Buttigieg in [7], [9] and were shown to exhibit properties akin to those of convolutional codes: they have a memory structure, which can naturally be represented via a trellis, and they are best suited for being decoded via a sequence maximum-likelihood (ML) or maximum a posteriori (MAP) Viterbi-like decoder (as opposed to decoding their codewords instantaneously). Furthermore, Buttigieg showed how the VLECs' distance spectrum and the union bound can be used to predict their error performance under hard-decision ML decoding for the binary symmetric channel (BSC) and identified the codes' free distance d_{free} as a key parameter which, when maximized, can improve the codes' performance. In related works, the error exponent of VLECs is analyzed [3] and conditions for the existence of VLECs are studied [31], [24].

In [7], Buttigieg originally proposed two techniques to construct VLECs with a given d_{free} value. They are respectively based on a greedy algorithm (GA) and a majority vote

Manuscript received August 2, 2012; revised January 17, 2013. The editor coordinating the review of this paper and approving it for publication was E. Ayanoglu.

T.-Y. Wu and P.-N. Chen are with the Department of Electrical and Computer Engineering, National Chiao-Tung University (NCTU), Hsinchu 300, Taiwan (e-mail: mavericktywu@gmail.com, poning@faculty.nctu.edu.tw). They are also with the Center of Information and Communications Technology of NCTU, Taiwan.

F. Alajaji is with the Department of Mathematics and Statistics, Queen's University, Kingston, ON K7L 3N6, Canada (e-mail: fady@mast.queensu.ca).

Y. S. Han is with the Department of Electrical Engineering, National Taiwan University of Science and Technology, Taipei 106, Taiwan (e-mail: yshan@mail.ntust.edu.tw).

This work was supported by the National Science Council of Taiwan under NSC 98-2221-E-009-060-MY3, NSC 99-2221-E-009-076-MY3, NSC 99-2221-E-011-158-MY3, and by the Natural Sciences and Engineering Research Council (NSERC) of Canada.

Parts of this work were presented at the IEEE International Symposium on Information Theory (ISIT 2011), Saint-Petersburg, Russia, July-August 2011 [32].

Digital Object Identifier 10.1109/TCOMM.2013.072913.120564

0090-6778/13$31.00 © 2013 IEEE

algorithm (MVA). Specifically, he employs either the GA or MVA procedure to select as many codewords as possible of the same length, where the selected codewords must satisfy certain minimum distance conditions in order to reach the required d_{free}. Later, Lamy and Paccaut [23] replaced Buttigieg's GA and MVA schemes with other ones in order to obtain a good trade-off between system complexity and coding efficiency. In [30], Wang *et al.* improved the coding efficiency of VLECs by iteratively replacing longer codewords with shorter ones. In [26], Savari and Kliewer 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 [11], [13], [18], Diallo *et al.* proposed several algorithms for obtaining VLECs with maximal d_{free} under the premise that all codeword lengths are known in advance. A similar approach was used in [12] for developing good error-correcting arithmetic codes.

With respect to VLEC decoding, Buttigieg [7] used a trellis representation of VLECs and modified the Viterbi algorithm (VA) to realize a sequence MAP decoder, which is optimal in terms of minimizing the VLECs' sequence error probability. Later in 2008, Huang *et al.* [19] proposed a trellis-based MAP priority-first search decoding algorithm for VLECs based on a suitable soft-decision MAP decoding criterion and empirically showed a significant complexity improvement over Buttigieg's MAP decoder. MAP decoding techniques using an extended trellis under the assumption that the receiver knows both the number of transmitted bits and the number of transmitted codewords were developed in [4], [21]. Other decoding methods for variable-length codes (VLC) that use other trellis VLC representations include the sequence MAP decoder of [3] and iterative (Turbo-like) decoders of [4], [22].

In this work, we present a novel priority-first search algorithm that can construct prefix-free VLECs with minimal average codeword length and free distance no less than a pre-given d^*_{free}. We next investigate how to select, among all obtained optimal[1] VLECs, the one with the best error correction capability. We observe that the codes' Levenshtein coefficient $B_{d_{\text{free}}}$ plays an important role in their error performance: choosing the optimal code with the smallest $B_{d_{\text{free}}}$ yields the best system error rate. Furthermore, we modify our construction algorithm to reduce its search complexity in order to accommodate large values of d_{free} and large source alphabets such as the 26-symbol English data source. We also propose a low-complexity two-phase sequence MAP decoder that can be applied to all VLECs (including our constructed optimal and suboptimal codes) under the assumption that the receiver knows both the number of transmitted bits and the number of transmitted codewords. We show by simulations that the resulting suboptimal VLECs outperform most existing VLECs in the literature in terms of compression efficiency, search complexity and error rate. We also compare our JSCC codes with traditional SSCCs.

The rest of this paper is organized as follows. In Section II, we formulate our problem and present some background material about VLECs. In Section III, we describe our code construction which guarantees the development of optimal VLECs with a given free distance constraint. In Section IV, two VLEC construction modifications are proposed respectively for the design of optimal codes with enhanced error correction capability and for the design of suboptimal VLECs for large d_{free} and large source alphabet sizes. In Section V, a low-complexity two-phase sequence MAP decoder is introduced. Simulation results illustrating the performance of the constructed optimal and suboptimal VLECs are given in Section VI. Finally, conclusions are stated in Section VII.

[1]We emphasize that, throughout the paper, an "optimal VLEC" is defined as a VLEC with minimal average codeword length. In other words, an optimal VLEC does not guarantee to yield the best error rate performance.

II. Problem Formulation and Preliminaries

We consider the JSCC problem of efficient compression of a discrete memoryless (independent and identically distributed) source and its reliable communication over a noisy channel via a single binary VLEC. We assume a binary phase-shift keying (BPSK) modulated additive white Gaussian noise (AWGN) channel (although other channel models can also be considered) and employ optimal sequence MAP decoding in the sense of minimizing the code's sequence error probability. The VLEC's free distance d_{free} has already been identified as a key error performance parameter, playing a similar role as for convolutional codes: the larger d_{free} is, the better is the code's error resilience particularly at high signal-to-noise ratios (SNRs) [7], [9]. Our objectives are four-fold:

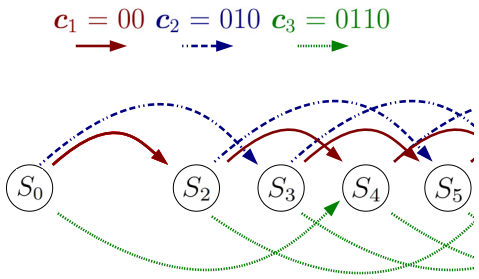
- Designing an algorithm that guarantees the construction of an optimal (i.e., with minimal average codeword length) binary prefix-free VLEC for a given free distance bound d^*_{free}.
- Enhancing the error correction capability of the constructed optimal VLECs by optimizing an important performance parameter $B_{d_{\text{free}}}$.
- Ensuring that the construction algorithms have a search complexity superior to the state-of-the-art code construction algorithms in the literature so that they can accommodate large source alphabets such as the 26-symbol English data source.
- Designing an efficient low-complexity sequence MAP decoder under the premise that the receiver knows the total number of transmitted VLEC codewords (in addition to the total number of transmitted code bits).

The successful achievement of these objectives has interesting applications for the effective compression and error-resilient transmission of text documents over noisy channels.

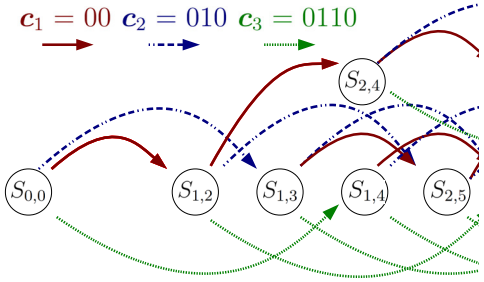
In what follows, we present some preliminary background about VLECs. Consider a K-ary discrete memoryless source with alphabet $\mathcal{S} \triangleq \{\alpha_1, \alpha_2, \ldots, \alpha_K\}$ and respective symbol probabilities $p_1, p_2, \ldots, p_K$ (such that $\sum_{i=1}^{K} p_i = 1$). A (first-order) VLEC encoder maps each symbol $\alpha_i \in \mathcal{S}$ to a binary variable-length codeword $\boldsymbol{c}_i$, where $i = 1, 2, \ldots, K$. The set of codewords is denoted by $\mathcal{C} = \{\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_K\}$ and the average codeword length for code $\mathcal{C}$ is given by $\overline{\mathcal{C}} \triangleq \sum_{i=1}^{K} p_i |\boldsymbol{c}_i|$, where $|\boldsymbol{c}_i|$ is the length of codeword $\boldsymbol{c}_i$.

A. Sequence MAP Decoding Criterion

Let $\mathcal{X}_{L,N} \triangleq \{\boldsymbol{x}_1\boldsymbol{x}_2\boldsymbol{x}_3\cdots\boldsymbol{x}_L : \forall \boldsymbol{x}_i \in \mathcal{C} \text{ and } \sum_{i=1}^{L} |\boldsymbol{x}_i| = N\}$ be a set of bitstreams consisting of L (concatenated)



(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 $\boldsymbol{c}_1$, $\boldsymbol{c}_2$ and $\boldsymbol{c}_3$.

codewords with overall length N. Define $\mathcal{X}_N \triangleq \bigcup_{i\geq 1} \mathcal{X}_{i,N}$. Assume that a sequence of VLEC codewords of overall length N is transmitted over the binary-input AWGN channel and that $\boldsymbol{r} \triangleq (r_1, r_2, \ldots, r_N)$ is received at the channel output. The sequence MAP (soft-decision) decoder then outputs $\hat{\boldsymbol{v}} \triangleq (\hat{v}_1, \hat{v}_2, \ldots, \hat{v}_N)$ if $\hat{\boldsymbol{v}}$ satisfies [19]

$$\sum_{i=1}^{N} (y_i \oplus \hat{v}_i)\|\phi_i\|_1 - \ln \Pr(\hat{\boldsymbol{v}}) \leq \sum_{i=1}^{N} (y_i \oplus v_i)\|\phi_i\|_1 - \ln \Pr(\boldsymbol{v}) \quad (1)$$

for all

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

where $\oplus$ is modulo-2 addition, $\Pr(\cdot)$ denotes probability, $\|\cdot\|_1$ denotes absolute value, ϕ_i is a log-likelihood ratio given by $\phi_i \triangleq \ln[\Pr(r_i|0)/\Pr(r_i|1)]$ and y_i is the hard decision of r_i given by

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

B. VLEC Trellis Diagrams

In [7], [9], Buttigieg employed a VLEC decoding trellis $\mathcal{T}_N$ as exemplified in Figure 1(a) for $\mathcal{C} = \{00, 010, 0110\}$, in which state S_j denotes that the number of bits decoded thus far is j.

We can construct an extended trellis $\mathcal{T}_{L,N}$ as defined in [4], [21] under the assumption that the receiver knows both L and N. An example of such extended trellis for $\mathcal{C} = \{00, 010, 0110\}$ is shown in Figure 1(b), where $S_{i,j}$ denotes that the number of decoded symbols and the number of decoded bits thus far are i and j, respectively.

C. Free Distance

In [7], in order to analyze the error performance of a trellis-based VLEC decoder, Buttigieg defined the free distance as the minimal Hamming distance between any two distinct paths which converge at the same node in the trellis. Thus, the free distance d_{free} of $\mathcal{C}$ as defined in [7] depends on the structure of its decoding trellis diagram. For the computation of d_{free}, we will assume throughout the paper that the receiver knows both L and N. Therefore, d_{free} is defined based on $\mathcal{X}_{L,N}$ and is given by

$$d_{\text{free}}(\mathcal{C}) \triangleq \min\{d(\boldsymbol{a}, \boldsymbol{b}) : \boldsymbol{a}, \boldsymbol{b} \in \mathcal{X}_{L,N} \text{ for some } L, N \text{ and } \boldsymbol{a} \neq \boldsymbol{b}\}, \quad (2)$$

where $d(\boldsymbol{a}, \boldsymbol{b})$ denotes the Hamming distance between bitstreams $\boldsymbol{a}$ and $\boldsymbol{b}$. The following lower bound on $d_{\text{free}}(\mathcal{C})$ can be shown [7], [9]

$$d_{\text{free}}(\mathcal{C}) \geq \min\{d_{\text{b}}(\mathcal{C}), d_{\text{c}}(\mathcal{C}) + d_{\text{d}}(\mathcal{C})\}, \quad (3)$$

where $d_{\text{b}}(\mathcal{C})$ is the "overall minimum block distance" defined as

$$d_{\text{b}}(\mathcal{C}) \triangleq \min\{d(\boldsymbol{c}_i, \boldsymbol{c}_j) : \boldsymbol{c}_i, \boldsymbol{c}_j \in \mathcal{C}, \boldsymbol{c}_i \neq \boldsymbol{c}_j \text{ and } |\boldsymbol{c}_i| = |\boldsymbol{c}_j|\}, \quad (4)$$

$d_{\text{c}}(\mathcal{C})$ is the "minimum converge distance" given by

$$d_{\text{c}}(\mathcal{C}) \triangleq \min\{d(\boldsymbol{c}_i, \boldsymbol{c}'_j) : \boldsymbol{c}_i, \boldsymbol{c}_j \in \mathcal{C}, |\boldsymbol{c}_i| < |\boldsymbol{c}_j|, \boldsymbol{c}'_j \text{ is the suffix of } \boldsymbol{c}_j \text{ and } |\boldsymbol{c}'_j| = |\boldsymbol{c}_i|\}, \quad (5)$$

and $d_{\text{d}}(\mathcal{C})$ is the "minimum diverge distance" defined as

$$d_{\text{d}}(\mathcal{C}) \triangleq \min\{d(\boldsymbol{c}_i, \boldsymbol{c}'_j) : \boldsymbol{c}_i, \boldsymbol{c}_j \in \mathcal{C}, |\boldsymbol{c}_i| < |\boldsymbol{c}_j|, \boldsymbol{c}'_j \text{ is the prefix of } \boldsymbol{c}_j \text{ and } |\boldsymbol{c}'_j| = |\boldsymbol{c}_i|\}. \quad (6)$$

III. Optimal VLEC Construction

We herein present a new search algorithm for constructing an optimal VLEC with a designed free-distance bound d^*_{free}. The search algorithm always outputs an optimal VLEC with its $d_{\text{free}} \geq d^*_{\text{free}}$. This algorithm, which is a modification and extension of the algorithm introduced in [20] for finding optimal lossless data compression codes with reversible VLC structure, uses a new search tree and a priority-first search method.

To construct an optimal VLEC with K codewords and $d_{\text{free}} \geq d^*_{\text{free}}$, we use a search tree in which each node $\mathtt{X}$ contains three components denoted by the triplet $\{\mathcal{C}_{\mathtt{X}}, \mathcal{A}_{\mathtt{X}}, f(\mathtt{X})\}$. Here, $\mathcal{C}_{\mathtt{X}} = \{\boldsymbol{c}^{\mathtt{X}}_1, \boldsymbol{c}^{\mathtt{X}}_2, \ldots, \boldsymbol{c}^{\mathtt{X}}_t\}$ denotes the set of t codewords that have been selected for the desired VLEC, and $\mathcal{A}_{\mathtt{X}} = \{\boldsymbol{a}^{\mathtt{X}}_1, \boldsymbol{a}^{\mathtt{X}}_2, \ldots\}$ is the set of all bitstreams, which can be future candidate codewords and hence do not contain any bitstreams that the codewords currently in $\mathcal{C}_{\mathtt{X}}$ are their prefixes. These bitstreams are listed in order of non-decreasing lengths:

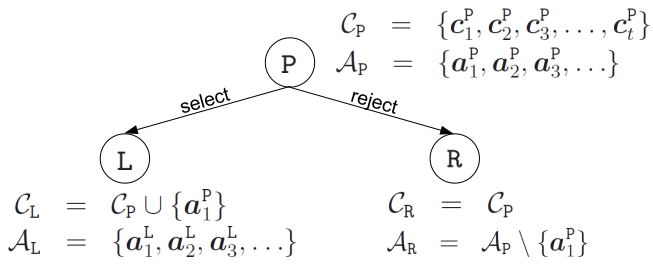


Fig. 2. Relation between a parent node and its children in a search tree.

$|\boldsymbol{a}_1^{\mathtt{X}}| \le |\boldsymbol{a}_2^{\mathtt{X}}| \le \cdots$.[2] Finally, $f(\mathtt{X})$ denotes the metric employed for finding an optimal VLEC and is given by

$$f(\mathtt{X}) \triangleq \sum_{i=1}^{t} p_i \cdot |\boldsymbol{c}_i^{\mathtt{X}}| + \sum_{i=t+1}^{K} p_i \cdot |\boldsymbol{a}_{i-t}^{\mathtt{X}}|. \tag{7}$$

The search tree is binary (i.e., each of its nodes except a leaf or terminal node has two children); the relation between a parent node and its children is illustrated in Figure 2. Specifically, for a parent node P, its left child L is obtained by adding the next candidate codeword $\boldsymbol{a}_1^{\mathtt{P}}$ into $\mathcal{C}_{\mathtt{L}}$. Since $\boldsymbol{a}_1^{\mathtt{P}}$ 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 $\boldsymbol{a}_1^{\mathtt{P}}$. Hence, the triplet of the left child L becomes

$$\mathcal{C}_{\mathtt{L}} = \mathcal{C}_{\mathtt{P}} \cup \{\boldsymbol{a}_1^{\mathtt{P}}\} \tag{8}$$

$$\begin{aligned}\mathcal{A}_{\mathtt{L}} &= \{\boldsymbol{a}_1^{\mathtt{L}}, \boldsymbol{a}_2^{\mathtt{L}}, \ldots\} \\ &= \{\boldsymbol{a} : \boldsymbol{a} \in \mathcal{A}_{\mathtt{P}} \text{ and } \boldsymbol{a}_1^{\mathtt{P}} \text{ is not a prefix of } \boldsymbol{a}\}\end{aligned} \tag{9}$$

$$\begin{aligned}f(\mathtt{L}) = &\sum_{i=1}^{t} p_i \cdot |\boldsymbol{c}_i^{\mathtt{P}}| + p_{t+1} \cdot |\boldsymbol{a}_1^{\mathtt{P}}| \\ &+ \sum_{i=t+2}^{K} p_i \cdot |\boldsymbol{a}_{i-t-1}^{\mathtt{L}}|.\end{aligned} \tag{10}$$

On the other hand, the right child R is obtained by rejecting the next candidate codeword $\boldsymbol{a}_1^{\mathtt{P}}$ from its parent node. So, the triplet of the right child R becomes

$$\mathcal{C}_{\mathtt{R}} = \mathcal{C}_{\mathtt{P}} \tag{11}$$

$$\mathcal{A}_{\mathtt{R}} = \{\boldsymbol{a}_2^{\mathtt{P}}, \boldsymbol{a}_3^{\mathtt{P}}, \ldots\} = \mathcal{A}_{\mathtt{P}} \setminus \{\boldsymbol{a}_1^{\mathtt{P}}\} \tag{12}$$

$$f(\mathtt{R}) = \sum_{i=1}^{t} p_i \cdot |\boldsymbol{c}_i^{\mathtt{P}}| + \sum_{i=t+1}^{K} p_i \cdot |\boldsymbol{a}_{i-t+1}^{\mathtt{P}}|. \tag{13}$$

Finally, since the root node has not yet selected any codeword, all bitstreams are its candidates; thus its components are given by

$$\mathcal{C}_{\mathtt{root}} = \emptyset \tag{14}$$

$$\begin{aligned}\mathcal{A}_{\mathtt{root}} &= \{\boldsymbol{a}_1^{\mathtt{root}}, \boldsymbol{a}_2^{\mathtt{root}}, \ldots\} \\ &= \{0, 1, 00, 01, 10, 11, 000, 001, \ldots\}\end{aligned} \tag{15}$$

$$f(\mathtt{root}) = \sum_{i=1}^{K} p_i \cdot |\boldsymbol{a}_i^{\mathtt{root}}|. \tag{16}$$

Since every possible VLEC can be obtained by traversing the search tree from the root node to its corresponding leaf nodes, a priority-first search algorithm can be applied on the tree to find a VLEC whose average codeword length is smallest among all VLECs with free distances no less than d_{free}^*. To reduce the search space, the average codeword length of any known VLEC with free distance no less than d_{free}^* is denoted by U_b and used as an upper bound for the average codeword length to exclude seemingly non-optimal VLECs during the search process. The search algorithm for finding an optimal VLEC is described as follows.

Step 1: Push the root node into the *Encoding Stack*.[3] Set upper bound U_b as the average codeword length of an existing VLEC with free distance no less than d_{free}^*.

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

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

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

1) It has selected more than K codewords for its $\mathcal{C}_{\mathtt{child}}$;
2) There is no more candidate in $\mathcal{A}_{\mathtt{child}}$ and the size of $\mathcal{C}_{\mathtt{child}}$ is less than K (i.e., $\mathcal{A}_{\mathtt{child}} = \emptyset$ and $|\mathcal{C}_{\mathtt{child}}| < K$);
3) The metric $f(\mathtt{child})$ is larger than U_b;
4) Its associated free distance $d_{\text{free}}(\mathcal{C}_{\mathtt{child}})$ is less than d_{free}^*.[4]

Step 5: Insert the remaining children (Those children which are not discarded in Step 4.) into the *Encoding Stack*, and reorder the *Encoding Stack* in order of ascending metrics. Go to Step 2.

It should be emphasized that the above construction algorithm focuses only on prefix-free VLECs as most previous works did [7], [9], [11], [12], [23], [26], [30]. Although non-prefix-free but uniquely decodable VLECs can also be constructed, they are not herein considered due to the added complexity in testing their unique decodability. The proof of the optimality of the above algorithm is provided in our conference work [32].

IV. Modified VLEC Constructions

In this section, two modifications on the optimal VLEC construction algorithm introduced in Section III are proposed. The first modification further enhances the error-correcting

[2]Recall that candidate codewords of equal length can be listed in any order without affecting the optimality of the output VLEC of our construction algorithm. For programing convenience, we simply list candidate codewords of equal length alphabetically in $\mathcal{A}_{\mathtt{X}}$, e.g., see $\mathcal{A}_{\mathtt{root}}$ in (15).

[3]The *Encoding Stack* can be implemented via the data structure named *HEAP* [10]. One important property of the *HEAP* structure is that it can access the node with the minimal metric (i.e., the top node in the *Encoding Stack*) within $O(\log(n))$ complexity, where n denotes the number of nodes in the *HEAP*.

[4]In order to check this condition efficiently, the lower bound on the free distance given in (3) is first computed; if it is less than d_{free}^*, then Dijkstra's algorithm [12] is adopted to determine the exact free distance. This is realized by transforming the finite-state VLEC encoder into a pairwise distance graph and applying Dijkstra's algorithm to find the graph's shortest path, where the resulting shortest path yields the VLEC's free distance. To our knowledge, Dijkstra's algorithm is the most efficient method to evaluate d_{free}.

capability of the found optimal VLEC by examining the union bound coefficient $B_{d_{\text{free}}}$ of all equivalent[5] optimal VLECs satisfying the free distance constraint and then outputting the one with the smallest $B_{d_{\text{free}}}$, where $B_{d_{\text{free}}}$ is a Levenshtein parameter defined in Section IV-A below. By targeting a suboptimal VLEC instead of an optimal one, the second modification reduces considerably the search complexity of the optimal construction algorithm in order to make feasible the construction of VLECs for larger alphabet sizes (such as the 26-symbol English data source) along with a large d^*_{free} (such as $d^*_{\text{free}} = 10$).

A. Finding an optimal VLEC with the smallest $B_{d_{free}}$

In [7], [9], Buttigieg found that under hard-decision ML decoding, the symbol error probability $P_e(\mathcal{C})$ of a VLEC $\mathcal{C}$ transmitted over the BSC with crossover probability ϵ can be upper-bounded by

$$P_e(\mathcal{C}) \leq \sum_{h=d_{\text{free}}(\mathcal{C})}^{\infty} \tilde{B}_h P_h, \tag{17}$$

where

$$\tilde{B}_h \triangleq \sum_{N=1}^{\infty} \sum_{\boldsymbol{a}\in\mathcal{X}_N} \Pr(\boldsymbol{a}) \cdot \left(\sum_{\boldsymbol{b}\,:\,\boldsymbol{b}\in\mathcal{X}_N \text{ and } d(\boldsymbol{a},\boldsymbol{b})=h} L(\boldsymbol{a},\boldsymbol{b}) \right) \tag{18}$$

and

$$P_h \triangleq \begin{cases} \sum_{e=(h+1)/2}^{h} \binom{h}{e} \epsilon^e (1-\epsilon)^{h-e} & \text{if } h \text{ is odd,} \\ \frac{1}{2}\binom{h}{h/2}\epsilon^{h/2}(1-\epsilon)^{h/2} & \\ \quad + \sum_{e=\frac{h}{2}+1}^{h} \binom{h}{e} \epsilon^e (1-\epsilon)^{h-e} & \text{if } h \text{ is even.} \end{cases} \tag{19}$$

Note that in Buttigieg's derivation, the symbol errors are counted using the Levenshtein distance $L(\cdot,\cdot)$ between transmitted sequence and decoded sequence, and the receiver decodes based on trellis $\mathcal{T}_N$ with N extending to infinity.

With a slight modification, a similar bound can be derived under the additional assumption that the receiver also knows the number of transmitted codewords L. In particular, (17) remains of the same form with $\tilde{B}_h$ replaced with B_h, where

$$B_h \triangleq \sum_{L=1}^{\infty} \sum_{N=1}^{\infty} \sum_{\boldsymbol{a}\in\mathcal{X}_{L,N}} \Pr(\boldsymbol{a}) \cdot \left(\sum_{\boldsymbol{b}\,:\,\boldsymbol{b}\in\mathcal{X}_{L,N} \text{ and } d(\boldsymbol{a},\boldsymbol{b})=h} L(\boldsymbol{a},\boldsymbol{b}) \right). \tag{20}$$

The coefficient B_h, as expressed above in (20), can be regarded as the average Levenshtein distance between all converging path pairs that are at a Hamming distance h from each other in the extended trellis $\mathcal{T}_{L,N}$. Thus, it is evident that B_h plays a key role in the union bound (17), particularly the first term $B_{d_{\text{free}}} \triangleq B_{h_{\min}}$, where $h_{\min}$ is the smallest integer h no less than $d_{\text{free}}(\mathcal{C})$ such that B_h is positive. Accordingly, given a set of optimal VLECs, the one with the smallest $B_{d_{\text{free}}}$ is expected to have a better error performance. It should be mentioned that in this paper we use a soft-decision MAP decoder with respect to the AWGN channel. The simplified union bound for the BSC however can provide a much simplified view on the system performance and hence the parameters $d_{\text{free}}(\mathcal{C})$ and $B_{d_{\text{free}}}$ obtained from (17) are adopted in our code design.[6]

We then modify the algorithm in Section III to find the optimal VLEC with the smallest $B_{d_{\text{free}}}$ among all optimal VLECs that has the minimum average codeword length. This can be achieved by continuing the algorithm, even if the top node of the *Encoding Stack* reaches the leaf node in Figure 2 (see Step 2 in Section III), until the average codeword length of the new top node is greater than that of the optimal VLEC. This continuation then guarantees that all optimal VLECs (of equal average codeword length) are examined and the one with the smallest $B_{d_{\text{free}}}$ can be selected. As a result, only the first two steps need to be modified:

Step 1′: Push the root node into the *Encoding Stack*. Set upper bound U_b as the average codeword length of an existing VLEC with free distance no less than d^*_{free}, and initialize $B^*_{d_{\text{free}}} = \infty$.

Step 2′: If the metric f (namely, the average codeword length) of the top node is strictly greater than U_b, then output $\mathcal{C}^*$ and stop the algorithm; else if the top node of the *Encoding Stack* has selected K codewords (i.e., $|\mathcal{C}_{\mathtt{top}}| = K$), and $d_{\text{free}}(\mathcal{C}_{\mathtt{top}}) \geq d^*_{\text{free}}$, and $B_{d_{\text{free}}}(\mathcal{C}_{\mathtt{top}}) < B^*_{d_{\text{free}}}$, then retain $\mathcal{C}^* = \mathcal{C}_{\mathtt{top}}$ and $B^*_{d_{\text{free}}} = B_{d_{\text{free}}}(\mathcal{C}_{\mathtt{top}})$. Delete the top node and reorder the *Encoding Stack* in order of ascending metrics.

B. Suboptimal code construction with parameters $(\Delta, \Gamma, \mathcal{D}, \mathcal{I})$

The complexity and memory demand of the optimal code construction algorithm in Section III grows significantly when searching for VLECs corresponding to a large source alphabet size K and a large free distance requirement d^*_{free}. We herein alleviate the algorithm's complexity and memory demand by constructing a suboptimal VLEC, which can accommodate higher free distance targets and larger source alphabet sizes. This is done based on four complexity reduction procedures.

First, we reduce the computational complexity incurred in examining the exact free distance of the top node by using its lower bound in (3) instead. Furthermore, Buttigieg recently observed [8] that good codes usually have converging and diverging distances (given in (5) and (6), respectively) that are equal (for even values of d_{free}) or differing by one (for odd values of d_{free}). Thus, we only focus on VLECs with the above property. In other words, the new suboptimal code construction only searches for the VLEC $\mathcal{C}$ that satisfies the following conditions:

$$\begin{cases} \min\{d_{\text{b}}(\mathcal{C}), d_{\text{c}}(\mathcal{C}) + d_{\text{d}}(\mathcal{C})\} \geq d^*_{\text{free}}, & \text{and} \\ |d_{\text{c}}(\mathcal{C}) - d_{\text{d}}(\mathcal{C})| \leq 1. \end{cases} \tag{21}$$

With this modification, the actual free distance of the output VLEC may be strictly larger than the required d^*_{free}; yet, this saves considerable computational effort in calculating the

[5]Two VLECs are said to be *equivalent* if they have identical average codeword length.

[6]We determine $B_{d_{\text{free}}}$ using the method proposed in [7, Section 3.5.1.1].

exact free distance for each node visited during the code search process.

Second, we adopt the early-elimination concept from [28], in which an efficient near-optimal sequential decoding algorithm for convolutional codes was proposed. In short, the authors in [28] propose to directly remove those nodes that are far behind the farthest node having been explored during the search process. Since the metric used in our code construction algorithm is also nondecreasing along every path in the trellis as in [28], these "far-behind" nodes are highly unlikely to result in a K-codeword offspring node whose average codeword length is small, and hence can be early-eliminated.

The third modification, also borrowed from [28], is to set a proper *Encoding Stack* size limitation in order to fix the memory demand and indirectly to reduce the search complexity.

In the last modification, we attempt to compensate for potential losses in coding efficiency (average codeword length) caused by the previous three modifications. Recall that the average codeword length of any existing VLEC can be used as the upper bound U_b in our search algorithm. Hence, when our suboptimal approach results in a VLEC whose average codeword length is smaller than the given U_b, we can update the value of U_b with this average codeword length and launch a new execution of our algorithm. This step can then be repeated in a number of iterations until no improvements in coding efficiency are realized or a prescribed maximal number of iterations is reached.

Four parameters $(\Delta, \Gamma, \mathcal{D}, \mathcal{I})$ are accordingly added corresponding to the last three modifications.

1: *Early elimination window* Δ: Ignore the top node in the *Encoding Stack*, whose number of codewords $|\mathcal{C}_{\mathtt{top}}|$ is less than $l_{\max} - \Delta$, where $l_{\max}$ is the largest $|\mathcal{C}|$ among all expanded nodes.

2: *Encoding Stack size* Γ: When the number of nodes in the *Encoding Stack* is larger than Γ, nodes are recursively deleted from the *Encoding Stack* according to one of the two criteria described below.

1) *Deletion criterion* $\mathcal{D} = \mathcal{D}_l$: Delete the node with the smallest code size $|\mathcal{C}|$.
2) *Deletion criterion* $\mathcal{D} = \mathcal{D}_m$: Delete the node with the largest metric f.

3: *The maximal number of iterations* $\mathcal{I}$.

The suboptimal algorithm, characterized by four parameters $(\Delta, \Gamma, \mathcal{D}, \mathcal{I})$, can thus be obtained by modifying the optimal algorithm in Section III and adding a new Step 6 as follows.

Step 1″: Push the root node into the *Encoding Stack*. Set upper bound U_b as the average codeword length of an existing VLEC with free distance no less than d^*_{free}. Alternatively for the followup iteration, set upper bound U_b as the average codeword length of the output VLEC obtained from the previous iteration. Initialize the target VLEC $\mathcal{C}^*$ as the empty set and $l_{\max} = 0$.

Step 2″: If the *Encoding Stack* is empty and $\mathcal{C}^* \neq \emptyset$, then output $\mathcal{C}^*$ as the optimal VLEC and stop the algorithm; else if both the *Encoding Stack* and $\mathcal{C}^*$ are empty, then report a code search failure and stop the algorithm.[7]
If $|\mathcal{C}_{\mathtt{top}}| < l_{\max} - \Delta$, then directly delete the top node from the *Encoding Stack* and redo Step 2″; else if $l_{\max} < |\mathcal{C}_{\mathtt{top}}|$, update $l_{\max} = |\mathcal{C}_{\mathtt{top}}|$.
If the top node of the *Encoding Stack* has selected K codewords (i.e., $|\mathcal{C}_{\mathtt{top}}| = K$) and $\mathcal{C}_{\mathtt{top}}$ satisfies condition (21), then output $\mathcal{C}_{\mathtt{top}}$ as the optimal VLEC and stop the algorithm.

Step 3″: Generate the two children of the top node as in Figure 2 and then delete the top node from the *Encoding Stack*. Then update U_b as the metric f of left child and put left child as $\mathcal{C}^*$ if left child satisfies all of the following conditions:

1) The left child has selected K codewords in his $\mathcal{C}_{\mathtt{left}}$;
2) $\mathcal{C}_{\mathtt{left}}$ satisfies condition (21);
3) Its associated metric f is smaller than U_b.

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

1) It has selected more than K codewords for its $\mathcal{C}_{\mathtt{child}}$;
2) There is no more candidate in $\mathcal{A}_{\mathtt{child}}$ and the size of $\mathcal{C}_{\mathtt{child}}$ is less than K (i.e., $\mathcal{A}_{\mathtt{child}} = \emptyset$ and $|\mathcal{C}_{\mathtt{child}}| < K$);
3) The metric $f(\mathtt{child})$ is larger than U_b;
4) It disobeys condition (21).

Step 5″: After inserting the remaining children into the *Encoding Stack*, recursively delete nodes from the *Encoding Stack* based on the chosen deletion criterion $\mathcal{D}$ until the *Encoding Stack* size is no greater than Γ. Reorder the *Encoding Stack* in order of ascending metrics. Go to Step 2″.

Step 6 : Repeat Steps 1″–5″ until either the maximum number of iterations $\mathcal{I}$ is reached or the upper bound U_b remains the same as the previous iteration.

We end this section by remarking on the free distances of the VLECs found by the three code construction algorithms introduced in this paper.

Recall that the two optimal code construction algorithms, respectively introduced in Sections III and IV-A, guarantee to output the VLEC whose average codeword length is smallest among all VLECs with free distance never smaller than the target free distance. In all cases we have examined, however, the free distance of the resulting optimal VLECs is always equal to the target free distance; although we conjecture the validity of this observation, we could not confirm it with a formal proof.

As expected, the suboptimal code construction algorithm

[7]Even if U_b is the average codeword length of an existing VLEC, the search space could be forced to become empty due to extra node exclusions of the first three complexity reduction modifications, i.e., requiring the free distance lower bound to be no less than d^*_{free}, early eliminations, and node deletions for a fully filled *Encoding Stack*. Note that when a node is excluded, all of its offspring nodes can no longer be visited; hence, it is possible that all the valid nodes (i.e., all the valid VLECs) are removed after several recursions of Steps 2″–5″.

Since, in the two earlier optimal code construction algorithms, the nodes corresponding to optimal VLECs will never be excluded, the *Encoding Stack* can never be empty prior to finding the optimal VLEC. Accordingly, it is not necessary to conduct an empty *Encoding Stack* check in these algorithms.

may produce a (suboptimal) VLEC with free distance strictly larger than d^*_{free}. However, in the particular case of the 26-symbol English alphabet (as will be presented in Section VI), the suboptimal code construction algorithm also consistently deliver a (suboptimal) VLEC with free distance equal to d^*_{free}, which indicates that the free distance lower bound in (3) is indeed tight for the found suboptimal VLEC. It should be mentioned that the tightness of (3) depends on the distribution of the source. In [13] and [18], it is shown that the tightness of (3) may be weak when the source distribution is highly unbalanced. Details will be given in Section VI.

V. Two-Phase Sequence MAP (TPSMAP) Decoding

In [19], an efficient sequence MAP decoder with the assumption that the receiver knows only the number of transmitted bits N was proposed. This decoder therefore can only operate on the traditional trellis $\mathcal{T}_N$ shown in Figure 1(a). With the additional information about the number of transmitted symbols L, we herein propose a new two-phase sequence MAP (TPSMAP) decoder, which can now operate on the extended trellis $\mathcal{T}_{L,N}$ (cf. Figure 1(b)), and whose average decoding complexity is only slightly greater than that for running the Viterbi algorithm (VA) on $\mathcal{T}_N$ (even if $\mathcal{T}_{L,N}$ has significantly more nodes and more transitions than $\mathcal{T}_N$). We next describe the TPSMAP decoding scheme.

In trellis $\mathcal{T}_{L,N}$, as defined in Section II-B and illustrated in Figure 1(b), a path traversing from $S_{0,0}$ to $S_{i,j}$ can be labeled as $\boldsymbol{x}_{(0,0)}^{(i,j)} \triangleq \boldsymbol{x}_1\boldsymbol{x}_2\cdots\boldsymbol{x}_i \in \mathcal{X}_{i,j}$, where each $\boldsymbol{x}_i \in \mathcal{C}$. Then, by following the MAP decoding criterion described in Section II-A, the path metric of $\boldsymbol{x}_{(0,0)}^{(i,j)}$ is defined as

$$\boldsymbol{g}\left(\boldsymbol{x}_{(0,0)}^{(i,j)}\right) = \sum_{\ell=1}^{j}(y_\ell \oplus b_\ell)\|\phi_\ell\|_1 - \ln \Pr\left(\boldsymbol{x}_{(0,0)}^{(i,j)}\right), \qquad (22)$$

where $b_1b_2\cdots b_j$ denotes the binary representation of path $\boldsymbol{x}_{(0,0)}^{(i,j)}$. Based on this new notation, the objective of the MAP decoder that knows both L and N is to find a path whose metric is the smallest among all valid paths $\boldsymbol{x}_{(0,0)}^{(L,N)}$ from $S_{0,0}$ to $S_{L.N}$.

In short, the TPSMAP scheme first performs backward VA on $\mathcal{T}_N$, whose size is significantly smaller than that of $\mathcal{T}_{L,N}$, and preserves the metric of each backward survivor path as $\boldsymbol{h}(S_j)$. The first phase of the TPSMAP is described as follows.

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

Step 2: Apply the backward VA with path metric given by (22) starting from S_N in $\mathcal{T}_N$, and record the metric and survivor path for each state as $\boldsymbol{h}(S_i)$ and $\boldsymbol{p}(S_i)$, respectively.

Step 3: If the number of codewords correspond to survivor path $\boldsymbol{p}(S_0)$ is equal to L, then output path $\boldsymbol{p}(S_0)$ as the MAP decision and stop the algorithm; otherwise, go to phase 2.

In the second phase, the TPSMAP applies a priority-first search algorithm [15] on $\mathcal{T}_{L,N}$ with the decoding metric of path $\boldsymbol{x}_{(0,0)}^{(i,j)}$ being re-defined as

$$\boldsymbol{m}\left(\boldsymbol{x}_{(0,0)}^{(i,j)}\right) = \boldsymbol{g}\left(\boldsymbol{x}_{(0,0)}^{(i,j)}\right) + \boldsymbol{h}\left(S_j\right). \qquad (23)$$

The second phase of the decoder is next described.

Step 1: Initialize the path metric of $\boldsymbol{x}_{(0,0)}^{(0,0)}$ as $\boldsymbol{m}(\boldsymbol{x}_{(0,0)}^{(0,0)}) = \boldsymbol{h}(S_0)$, and load it into the *Decoding Stack*.[8]

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

Step 3: Mark the state of the top node as visited. Then extend the top node to all its successors and compute their metrics according to (23). Delete the top node from the *Decoding Stack*.

Step 4: Discard the successors if they had been marked as visited. Also, 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 (those successors which are not discarded in Step 4) into the *Decoding Stack* and reorder the *Decoding Stack* in order of ascending $\boldsymbol{m}$-metrics defined in (23). Go to Step 2.

It can be noted that the second phase of the decoder follows similar procedures as the code construction algorithm introduced in Section III, except that the priority-first algorithm is now applied on the trellis $\mathcal{T}_{L,N}$ instead of applying it on a search tree for code construction. Since some paths of the trellis $\mathcal{T}_{L,N}$ run across the same node, the priority-first algorithm must avoid expanding the same node on the trellis $\mathcal{T}_{L,N}$ more than once. We therefore need to mark the expanded node (top node) as visited in Step 3, and discard the successors which have already been marked as visited in Step 4. The proof of optimality for the above decoding algorithm is presented in our conference work [32].

VI. Simulation Results

In this section, we assess via simulations the error performances of the found VLECs in terms of reconstructed source symbol error rate (SER).[9] In all simulations, the source is assumed memoryless and the channel is the BPSK-modulated AWGN channel. The decoding complexity of the proposed two-phase sequence MAP (TPSMAP) decoder is also examined. Furthermore, comparisons with other systems in literature, including three known VLEC schemes and a traditional SSCC system, are provided. For measuring the time to search for the optimal and suboptimal VLECs, the experiments were carried using the *C* programing language under a *64-bit* operation system *Linux (Ubuntu 10.04 LTS)* executed on a desktop computer with a *Intel-Core2 Duo E6600 2.4GHz CPU* and *4GB* memory.

As usual, the system signal-to-noise ratio (SNR) is given by SNR $\triangleq E/N_0$, where E is the signal energy per channel use and $N_0/2$ is the variance of the zero-mean additive channel noise sample. To account for the coding redundancy

[8]The role of the *Decoding Stack* is similar to that of the *Encoding Stack*, except that the *Decoding Stack* stores the nodes of $\mathcal{T}_{L,N}$ as its elements. It is also implemented via the data structure named *HEAP* [10] and accesses the node with minimal metric (i.e., its top node) within $O(\log(n))$ complexity, where n denotes its total number of nodes.

[9]As a convention, the SER here is the Levenshtein distance between the transmitted sequence and the decoded sequence divided by the number of transmitted source symbols (i.e., L).

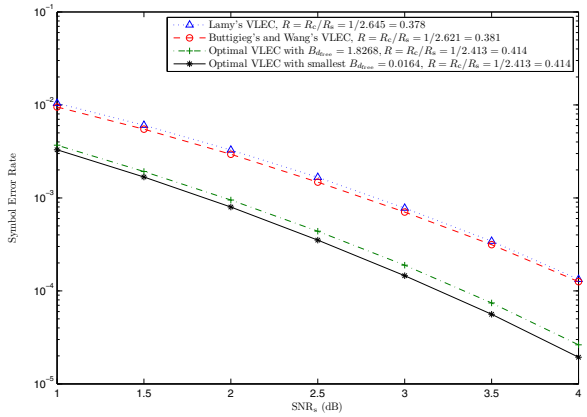


Fig. 3. Error performances of different (3rd order) VLECs for a binary non-uniform source with $p_0 = 0.8$. The number of 3-bit source symbols per transmission block is 10, which is equivalent to 30 source information bits. The free distance d_{free} for all VLECs is $d_{\text{free}} = 7$.

of systems with different code rates, SNR per source symbol is used in presenting the simulation results, which is given by

$$\text{SNR}_\text{s} = \frac{E_\text{s}}{N_0} = \frac{E}{N_0} \cdot \frac{1}{R}, \tag{24}$$

where E_s is the energy per source symbol, and R is the overall (average) system rate defined as the number of transmitted source symbols per channel use. For an SSCC system, the overall rate R satisfies $R = R_\text{c}/R_\text{s}$, where R_s is the source coding rate (in coded bits/source symbol) and R_c is the channel coding rate (in coded bits/channel use). Hence, for an SSCC system employing a kth-order Huffman VLC[10] followed by a tail-biting convolutional code, R_s is the average codeword length of the Huffman code divided by k, and R_c is the rate of the tail-biting convolutional code. Note that a VLEC (or a single-step JSCC) can be regarded as having $R_\text{c} = 1$ with R_s being its averaged source coding rate, since no explicit channel coding is performed.

We first examine in Figure 3 the improvement in error performance between the optimal code construction in Section III and the modified optimal one (that guarantees to output the optimal VLEC with the smallest $B_{d_{\text{free}}}$) in Section IV-A. Here, we group three information bits, generated from a binary non-uniform memoryless source with bit probability $p_0 \triangleq \Pr(0) = 0.8$, as one source symbol; hence, the VLECs are 3rd order VLCs (i.e., $k = 3$), and the size of the source alphabet is $K = 2^3 = 8$. Also shown in the same figure are the error performances of three VLECs respectively obtained by Buttigieg's [7], Lamy's [23] and Wang's [30] code construction algorithms, which have the same free distance $d_{\text{free}} = 7$ as the optimal VLECs we constructed, where Buttigieg's and Lamy's algorithms coincidentally yield an identical code in this case. In each simulation, 10 source symbols (equivalently, 30 source information bits) are encoded and transmitted as a block. All codes are decoded using the TPSMAP decoder of Section V. Figure 3 shows that our optimal VLEC constructed by the algorithm proposed in Section III has around 0.8 dB coding gain over the three existing VLECs; it also indicates that minimizing $B_{d_{\text{free}}}$ can further pick up another 0.1 dB in performance gain.

Table I summarizes the decoding complexity of the TPSMAP for the VLECs of Figure 3. We notice that a VLEC with higher average codeword length requires a higher decoding complexity. This is somehow anticipated since the decoding trellis is larger for a VLEC with higher average codeword length. Along this observation, the optimal VLEC and the optimal VLEC with the smallest $B_{d_{\text{free}}}$ have expectedly similar decoding complexity because they have identical average codeword length. In addition, with a smaller (actually, the minimum) average codeword length, our optimal VLEC decodes faster via the TPSMAP than the other three VLECs.

We next test the performance of the suboptimal code construction algorithm of Section IV-B for the 26-symbol English data source. Since there are two different distributions for the English alphabet that are generally used in the literature for constructing VLECs (e.g., compare [25], [30], [20], [26] with [7], [9], [13], [18]), we provide simulation results for both distributions; we will refer to them as Distributions 1 and 2, respectively. The VLECs we obtain via our suboptimal code construction algorithm are presented in Tables II and III for Distributions 1 and 2, respectively.

In Table IV(a), we list, for different values of d_{free}, the average codeword lengths (ALs) of the resulting VLECs under Distribution 1 as well as the execution time needed for their construction via our suboptimal algorithm and the three algorithms referred above. For the sake of completeness, the parameters used in each algorithm are reported in Table IV(b).[11] These parameters are chosen through a number of trials in targeting a VLEC with smaller average codeword length. The results indicate that by manipulating the parameters, the VLECs obtained by our suboptimal code construction algorithm can outperform all other three VLECs in average codeword length. Table IV(a) also shows that our suboptimal code construction algorithm is worse than Lamy's or Wang's algorithms in terms of execution time for $d_{\text{free}} \leq 9$; however, we can prevent the construction complexity of our algorithm from growing too quickly for $d_{\text{free}} \geq 10$ by properly adjusting its parameters under the premise that our algorithm can still yield a better code than the other three algorithms. Similar conclusions can be drawn about the performance of the above algorithms under Distribution 2; the results are presented in Table V.

Analogously to other schemes, many combinations of parameters need to be tested in our suboptimal algorithm to arrive at a good VLEC construction. The main parameters that control the algorithm's complexity are the early-elimination window Δ and the *Encoding Stack* size Γ. Usually, complexity increases when either Δ or Γ increase, albeit with the benefit of improving the VLEC average codeword length. In general, it is not straightforward to decide on the right choice of values for these parameters before testing them. Despite this inconvenience, the proposed suboptimal approach is efficient enough to test many combinations of parameters

[10]Recall that a kth order VLC maps a block of k source symbols onto a variable-length codeword. So its average source coding rate is given by the average codeword length divided by k.

[11]Buttigieg's algorithm (specifically, MVA in [7]) and Wang's algorithm [30] are characterized by two parameters, L_1 and $L_{\max}$. An additional parameter L_s is needed for Lamy's algorithm (specifically, noHole+L_s in [23]).

TABLE I
AVERAGE (AVG) AND MAXIMUM (MAX) NUMBERS OF DECODER BRANCH METRIC COMPUTATIONS AS WELL AS THEIR AVERAGE CODEWORD LENGTHS (ALS) PER GROUPED SYMBOL FOR THE CODES OF FIGURE 3.

SNR_s		1 dB		2 dB		3 dB		4 dB	
VLEC system	AL	AVG	MAX	AVG	MAX	AVG	MAX	AVG	MAX
Lamy's VLEC	7.936	511	3631	510	1858	510	970	510	731
Buttigieg's and Wang's VLECs	7.864	500	3439	499	1303	499	720	499	670
Optimal VLEC	7.240	461	2970	460	1119	459	719	459	668
Optimal VLEC with smallest $B_{d_{\text{free}}}$	7.240	462	3040	460	1144	459	712	459	668

TABLE II
THE VLECS FOR THE ENGLISH ALPHABET WITH DISTRIBUTION 1 OBTAINED BY THE SUBOPTIMAL CODE CONSTRUCTION ALGORITHM FOR DIFFERENT VALUES OF FREE DISTANCE.

Alphabet	Probability	$d_{\text{free}}=3$	$d_{\text{free}}=5$	$d_{\text{free}}=7$	$d_{\text{free}}=9$	$d_{\text{free}}=10$	$d_{\text{free}}=11$
E	0.14878610	0111	00001	00000000	00101101	000100000	0000000000
T	0.09354149	00101	01110	11111111	111111100	0000011110	00001011111
A	0.08833733	11011	0101011	000011111	1111000111	00101100111	000111101001
O	0.07245769	000110	1010000	111100001	11001000100	11011011000	001101010111
R	0.06872164	010011	00110100	0011010100	110001111011	010111101100	00111100111001
N	0.06498532	101111	10010011	1100110011	0101010010100	101010010011	11101011100110
H	0.05831331	111010	11101111	01011010010	1001001100011	0110111000010	010101110010101
I	0.05644515	0001011	011001011	10101010101	00010000010001	111110011101	111011101111010
S	0.05537763	1000100	101111100	11000101001	10100010101010	10110011110101	0111110110110011
D	0.04376834	1011001	110000100	001111001100	00110010010000	11001100001011	1100011111011100
L	0.04123298	1110010	10111101111	010101100010	100000110110011	011010110110011	011011011001011010
U	0.02762209	00000011	1101000010	101010010001	000110101011011	100111011101111	11010010111110110
P	0.02575393	00000100	11000100111	110000111100	010001101100101	111101101010100	101001001101110100
F	0.02455297	10001111	11110101000	0101001110110	100000000110000	011000110111001	110101111010111100
M	0.02361889	10010101	110001010011	0110011000011	010000100111101111	101111010000101	111010000110010110
C	0.02081665	10100001	110111001100	0110100111001	010110111010110001	010010011101110110	011100101111110010100
W	0.01868161	10100110	1100010101000	1001011011001	100001101100010010	10100010110010001	011110001011111010011
G	0.01521216	11000000	1101110000010	1001110000110	000101101101110010	11010101111101001	011001111011111000011
Y	0.01521216	010000011	11011101010111	1010001101100	010000110101010101	010010011101010011	110101000101111010010110
B	0.01267680	010000100	1101110101000	0011001111010010	100101111010001000	110000101110001100	101001101011111111000101
V	0.01160928	100100000	110111000010111	0101100110111	000101010110101010001	111111100111111010	111011000000011110100010
K	0.00867360	110001101	1101110101111001	0110010111100	010000101110101010	100011010100100110	010001101100111111000011
X	0.00146784	1000001001	1101110110001100	0110110000011	010000100000010000	110010101111000011	101110010000111111010010
J	0.00080064	1100001111	1101110001111001	1001100101100	000101101111101100	111001000101110110	111011000001000110110110
Q	0.00080064	1100011100	1101110101100100	1010011001010	000110101100100111	111000100110110001	010101010100011111000011
Z	0.00053376	10000010100	1101110110111111	0011011011111001	010000110100110010000	111001011101101011	101011001000100111010010

TABLE III
THE VLECS FOR THE ENGLISH ALPHABET WITH DISTRIBUTION 2 OBTAINED BY THE SUBOPTIMAL CODE CONSTRUCTION ALGORITHM FOR DIFFERENT VALUES OF FREE DISTANCE.

Alphabet	Probability	$d_{\text{free}}=3$	$d_{\text{free}}=5$	$d_{\text{free}}=7$	$d_{\text{free}}=9$	$d_{\text{free}}=10$	$d_{\text{free}}=11$
E	0.1270	0111	000000	0011111	00000101	000100000	0000000001
T	0.0906	00011	111111	01000110	001110011	0000011110	00001111101
A	0.0817	11101	0001110	000010000	0101101000	00101100111	01110001010
O	0.0751	001010	1111000	111101101	01110111001	11011011000	01110111010
I	0.0697	010011	00101001	0001001001	001111100110	010111101100	10110110111000
N	0.0674	101111	11010110	1110111000	110010011000	101010010011	11011101000111
S	0.0633	110110	010110100	00000101100	010011101111	01101110000010	101100101001110
H	0.0609	0010010	101100110	10001110011	01110110110110	1111100111101	111011010110000
R	0.0599	0100000	110010011	11110000001	10001011011100	10110011110101	101110011101010011
D	0.0425	1000110	111001101	010110100010	100011100011010	11001100001011	1110011000101100
L	0.0403	1011001	0100010101	100111010001	110100010101110	011010110110011	110110101100101000
C	0.0278	1101011	0101001011	101001111010	0000101011001010	100111011101111	110110100010011110
U	0.0276	10001011	10001100010	111000100101	101011110011110	111101101010100	111011101101100010
M	0.0241	10010100	1010011001	000100111010101	110101111101000	011000110111001	101110110111000000
W	0.0236	10100001	1011100101	1000011101010	010100001100100 10	101110110000101	110110111000101111
F	0.0223	10100110	01001010101	1100100100001	100111111110011	010010011101101 10	111001001001010010
G	0.0202	11000010	01011001011	001000101000011	111110110101111 10	101000101100100 01	101110110000101011 10
Y	0.0197	11000101	10001100011	101100111101000	010111111101011 10	110101011111011001	111010101111011100
P	0.0193	000000100	10110100101	110011111100 11	100111001010001 0 10	010010011101010011	111001001001011110 110
B	0.0149	100000001	011001000111	11011000101010	111000010011001110	011101110110101011	111010110000101101010
V	0.0098	100001111	100001010011	001001000100011	001011101001100 1110	111000001010001101	111110100111111011101
K	0.0077	100100010	111010100011	011011001111100	100100001101010101 0	100011010100100 1101	101110110000011000011 0
J	0.0015	0000001111	0011111100011	100111011100000	111001101111001001 0	101101110101111101 0	111010110111111101101
X	0.0014	0000011010	001001111000 11	0010001011100000 0	011010000111110010 11	111000100101111010 1	101110110110111111101100
Q	0.0010	00000111010	11000010100011	110100001111010 0	101000000110101111 10	111001110110110001 11	111010001001010001011 1
Z	0.0007	000001110010	011101111100011	1110001100100011	110111011110110011 10	111111001001101110 01	111110100011000101110 10

in reasonable time. For example, to get the suboptimal VLEC with $d_{\text{free}} = 3$ in Table II, we simulated all combinations of the following parameters: $\Delta = \{1, 3, 5, 7, 9, 11, 13\}$, $\Gamma = \{20, 40, 60, 80, 100, 200, 300, 400, 500, 1000\}$ and $\mathcal{D} = \{\mathcal{D}_m, \mathcal{D}_l\}$. It took us only about 29 minutes to simulate all these 140 combinations within a single computer experiment.

We next provide efficiency comparisons with the recent works of Diallo *et al.* [13] and Hijazi *et al.* [18]. [12] Notably different from our work and also the main referenced works in this paper (i.e., Buttigieg's [7], Lamy's [23] and Wang's [30]), Diallo *et al.* and Hijazi *et al.* do not construct codes for a given distribution but for a pre-specified set of codeword lengths. The distributions assumed in their papers are therefore primarily for the computation of the resulting average codeword length. To compare with the VLEC of [13], we simulated our suboptimal code construction for $d_{\text{free}} = 4$ under the same used distribution for the 26-symbol English alphabet (Distribution 2 given in Table III). The VLEC designed in [13, Table IV] has an average codeword length of 7.3375 and an execution time of 310 hours. Our suboptimal code construction algorithm, when initialized by an upper bound given by the average length of the code in [13] (i.e., with $U_b = 7.3375$) and parameters $(\Delta = 3, \Gamma = 200, \mathcal{D}_l, \mathcal{I} = 1)$ yields an improved VLEC with an average codeword length of 6.4794 within only

[12]It should be mentioned that we did not actually implement the systems of [13] and [18]; instead, the efficiency results of these systems are directly retrieved from each paper. Due to differences in the experimental platforms, the comparisons between our system and those of [13] and [18], especially in terms of execution time, may not be on a fully equal footing. They are however herein provided for the sake of reference.

TABLE IV

LIST OF THE VLECS OBTAINED BY THREE EXISTING CODE CONSTRUCTION SCHEMES AND THE VLECS OBTAINED BY OUR SUBOPTIMAL CODE CONSTRUCTION ALGORITHM FOR THE 26-SYMBOL ENGLISH ALPHABET WITH DISTRIBUTION 1 GIVEN IN TABLE II: (a) AVERAGE CODEWORD LENGTHS (ALS) OF THE FOUND CODES AND EXECUTION TIME FOR EACH CODE CONSTRUCTION ALGORITHM; (b) PARAMETERS USED IN EACH ALGORITHM. THE SUBOPTIMAL ALGORITHM IS INITIALIZED WITH U_b SET TO EQUAL THE SMALLEST OF THE AVERAGE CODEWORD LENGTHS OF THE VLECS BY BUTTIGIEG, LAMY AND WANG.

(a)

Algorithm	Buttigieg's		Lamy's		Wang's		Suboptimal	
	AL	Time	AL	Time	AL	Time	AL	Time
$d_{\text{free}} = 3$	6.272617	2m2s	6.309980	4s	6.266612	<1s	6.189350	18s
$d_{\text{free}} = 5$	8.378035	6m42s	8.400986	44s	8.378035	12s	8.333866	2m27s
$d_{\text{free}} = 7$	10.559646	4h31m	10.599945	5m43s	10.488923	27s	10.302508	8m41s
$d_{\text{free}} = 9$	12.737255	6h27m	12.806644	9m52s	12.737255	2m30s	12.532291	5m29s
$d_{\text{free}} = 10$	12.757672	11h45m	12.867893	17m54s	12.757672	47m46s	12.593140	9m35s
$d_{\text{free}} = 11$	14.876166	19h14m	15.354549	21m43s	15.024952	2h15m	14.580329	14m53s

(b)

Algorithm	Buttigieg's	Lamy's	Wang's	Suboptimal
Parameters	$(L_1, L_{\max})$	$(L_1, L_{\max}, L_s)$	$(L_1, L_{\max})$	$(\Delta, \Gamma, \mathcal{D}, \mathcal{I})$
$d_{\text{free}} = 3$	$(4, 13)$	$(4, 13, 10)$	$(4, 13)$	$(5, 300, \mathcal{D}_m, 2)$
$d_{\text{free}} = 5$	$(6, 15)$	$(6, 15, 12)$	$(6, 15)$	$(3, 500, \mathcal{D}_l, 1)$
$d_{\text{free}} = 7$	$(7, 16)$	$(7, 16, 13)$	$(7, 16)$	$(5, 2000, \mathcal{D}_m, 1)$
$d_{\text{free}} = 9$	$(9, 18)$	$(9, 18, 15)$	$(9, 18)$	$(1, 60, \mathcal{D}_m, 1)$
$d_{\text{free}} = 10$	$(10, 19)$	$(10, 19, 15)$	$(10, 19)$	$(1, 40, \mathcal{D}_l, 2)$
$d_{\text{free}} = 11$	$(12, 21)$	$(12, 21, 17)$	$(12, 21)$	$(1, 4, \mathcal{D}_l, 1)$

TABLE V

LIST OF THE VLECS OBTAINED BY THREE EXISTING CODE CONSTRUCTION SCHEMES AND THE VLECS OBTAINED BY OUR SUBOPTIMAL CODE CONSTRUCTION ALGORITHM FOR THE 26-SYMBOL ENGLISH ALPHABET WITH DISTRIBUTION 2 GIVEN IN TABLE III: (a) AVERAGE CODEWORD LENGTHS (ALS) OF THE FOUND CODES AND EXECUTION TIME FOR EACH CODE CONSTRUCTION ALGORITHM; (b) PARAMETERS USED IN EACH ALGORITHM. THE SUBOPTIMAL ALGORITHM IS INITIALIZED WITH U_b SET TO EQUAL THE SMALLEST OF THE AVERAGE CODEWORD LENGTHS OF THE VLECS BY BUTTIGIEG, LAMY AND WANG.

(a)

Algorithm	Buttigieg's		Lamy's		Wang's		Suboptimal	
	AL	Time	AL	Time	AL	Time	AL	Time
$d_{\text{free}} = 3$	6.4038	20s	6.4047	14s	6.3574	<1s	6.2560	7s
$d_{\text{free}} = 5$	8.4740	5m16s	8.5049	47s	8.4740	9s	8.3223	1m13s
$d_{\text{free}} = 7$	10.5388	1h55m	10.5110	12m01s	10.5388	47s	10.3615	12m13s
$d_{\text{free}} = 9$	12.8898	3h14m	12.9644	13m04s	12.8898	4m22s	12.6647	6m03s
$d_{\text{free}} = 10$	12.8959	9h10m	13.0095	58m29s	12.8959	19m41s	12.7507	8m49s
$d_{\text{free}} = 11$	15.0345	17h37m	15.0846	38m53s	15.0345	1h20m	14.6521	16m12s

(b)

Algorithm	Buttigieg's	Lamy's	Wang's	Suboptimal
Parameters	$(L_1, L_{\max})$	$(L_1, L_{\max}, L_s)$	$(L_1, L_{\max})$	$(\Delta, \Gamma, \mathcal{D}, \mathcal{I})$
$d_{\text{free}} = 3$	$(4, 13)$	$(4, 13, 13)$	$(4, 13)$	$(6, 200, \mathcal{D}_m, 1)$
$d_{\text{free}} = 5$	$(6, 15)$	$(6, 15, 13)$	$(6, 15)$	$(2, 250, \mathcal{D}_m, 1)$
$d_{\text{free}} = 7$	$(7, 18)$	$(7, 18, 15)$	$(7, 18)$	$(1, 3000, \mathcal{D}_m, 1)$
$d_{\text{free}} = 9$	$(9, 18)$	$(9, 18, 16)$	$(9, 18)$	$(1, 20, \mathcal{D}_l, 1)$
$d_{\text{free}} = 10$	$(10, 20)$	$(10, 20, 17)$	$(10, 20)$	$(3, 40, \mathcal{D}_l, 1)$
$d_{\text{free}} = 11$	$(11, 21)$	$(11, 21, 18)$	$(11, 21)$	$(1, 12, \mathcal{D}_l, 1)$

2 seconds of execution. [13]

In [18, Table 3], Hijazi *et al.* provide a VLEC for $d_{\text{free}} = 7$ within an execution time of 13 minutes and 31 seconds for a given set of codeword lengths. For Distribution 2 in Table III, the resulting average codeword length is 10.4213. In [18, Table 4], they provide another VLEC for $d_{\text{free}} = 7$, resulting in a better average codeword length of 10.1138 under Distribution 2, but no execution time is given.

In contrast, our best to-date suboptimal code construction, as shown in Table V with parameters $(\Delta = 1, \Gamma = 3000, \mathcal{D}_m, \mathcal{I} = 1)$ and $U_b = 10.5110$, outputs a VLEC for $d_{\text{free}} = 7$ with an average codeword length of 10.3615, which is in between 10.4213 [18, Table 3] and 10.1138 [18, Table 4], under an execution time of 12 minutes and 13 seconds. On the other hand, our current suboptimal code construction algorithm, when initialized with $U_b = 10.4213$ (and also $U_b = 10.1138$), either reports a code search failure or cannot converge to a solution in reasonable time, depending on the choice of parameters $(\Delta, \Gamma, \mathcal{D}, \mathcal{I})$. It should be pointed out however, that unlike our suboptimal algorithm, the scheme of [18] requires a priori knowledge of all codeword lengths before it is run. Hence arriving at the right choice of codeword lengths for any given d_{free} and alphabet size requires additional trials (whose execution duration are not reported in [18]). Nonetheless, it is certainly of interest, to further improve the

[13] We have noticed from recent results, that even without making use of the U_b parameter (i.e., by setting $U_b = \infty$), the (3,200,$\mathcal{D}_l$,1) suboptimal algorithm still yields a VLEC with an average codeword length of 6.4794 within only 2 seconds of execution. This further shows that the proposed suboptimal algorithm is highly efficient.

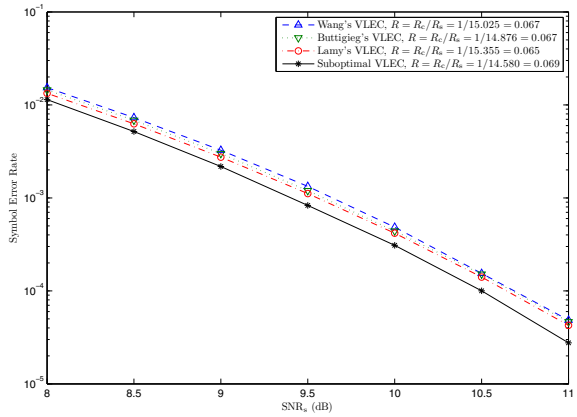


Fig. 4. Error performances of the VLECs of Table IV with $d_{\text{free}} = 11$ for the 26-symbol English alphabet (with Distribution 1). The number of source symbols per transmission block is $L = 10$.

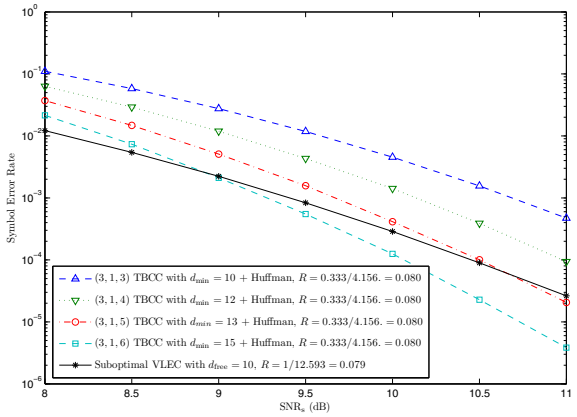


Fig. 5. Error performances of the SSCC (specifically, first order Huffman + TBCC) and the VLEC of Table IV with $d_{\text{free}} = 10$ for the 26-symbol English alphabet (with Distribution 1). The number of source symbols per transmission block is $L = 10$.

efficiency of our algorithm and assess whether or not the average codeword length of 10.1138 is optimal or not for $d_{\text{free}} = 7$.

Figure 4 illustrates the SER performances of the VLECs presented in Table IV with $d_{\text{free}} = 11$. Again, 10 source symbols are encoded and transmitted as a block in each simulation, and all codes are decoded using the TPSMAP decoder in Section V. We observe from the figure that the VLEC obtained by our suboptimal code construction algorithm outperforms the other three VLECs by at least 0.15 dB. The decoding complexities of these systems are summarized in Table VI. As anticipated, the VLEC obtained by our suboptimal code construction algorithm has the smallest average codeword length and hence its decoding complexity is smaller than those of the other three VLECs, particularly in the maximum number of branch metric computations.

Finally, we compare the SER performance of one suboptimal VLEC shown in Table IV with that of a traditional SSCC system for the situation where the source is the memoryless 26-symbol English data. The SSCC system consists of a Huffman source coder and a tail-biting convolutional channel (TBCC) coder. We use $(3, 1, 3)$, $(3, 1, 4)$, $(3, 1, 5)$ and $(3, 1, 6)$ TBCCs respectively with generator polynomial $[54, 64, 74]$, $[52, 66, 76]$, $[47, 53, 75]$ and $[564, 624, 754]$ (in octal) [29] such that the resulting SSCC systems have approximately the same code rate $R \approx 0.08$ as the VLEC to be compared with. Also, the d_{free} of the chosen VLEC is 10, while the largest minimum Hamming distances $d_{\min}$ for $(3, 1, 3)$, $(3, 1, 4)$, $(3, 1, 5)$ and $(3, 1, 6)$ TBCCs are 10, 12, 13 and 15, respectively. Both the VLEC and the TBCCs are decoded by sequence decoders, where the one for the VLEC is the TPSMAP proposed in Section V, and the one for the TBCCs is the priority-first search decoding algorithm (PFSA) introduced in [16]. The results are illustrated in Figure 5.

We remark from Figure 5 that for almost all simulated SNRs, the suboptimal VLEC outperforms the SSCC using a TBCC of memory order no larger than 5. In comparison with the SSCC equipped with the $(3, 1, 6)$ TBCC, the suboptimal VLEC still performs better when $\mathrm{SNR_s}$ is less than 9 dB. Table VII summarizes the decoding complexities of the suboptimal VLEC and the TBCCs in terms of the branch metric computations. It indicates that the VLEC system is more efficient than the SSCC system using a TBCC of memory orders 5 and 6. Note that in this table, the decoding complexity of the Huffman coder is not even included. We can then conclude that the VLEC system can achieve a better performance than an SSCC system of comparable decoding complexity. We end the discussion by pointing out again that the VLEC system only requires one encoder and one decoder, while the SSCC system needs separate source coder and channel coder at both transmitter and receiver sides. This can be considered another advantage of the VLEC system over the SSCC system.

VII. Conclusion

In this work, a novel search algorithm is proposed for constructing optimal prefix-free VLECs for the effective joint source-channel coding of memoryless sources over memoryless channels. The optimal construction algorithm is modified to construct optimal VLECs with improved resilience against channel noise through a critical union bound parameter $B_{d_{\text{free}}}$. A suboptimal but much more efficient construction algorithm is next presented to construct VLECs with large free distances and for large source alphabets such as the 26-symbol English data source. A low-complexity two-phase sequence MAP (TPSMAP) decoder for the VLECs is also proposed. Simulations show that the developed optimal and suboptimal VLECs can have evident gains over most existing VLECs of identical free distance in terms of average codeword length, error rate performance and decoding complexity. Also shown in this paper is that our VLEC system outperforms traditional separate source/channel coding systems of similar overall rate at low to medium SNRs with the benefit of considerably smaller decoding complexity. Future research directions may include further improving the efficiency of our sub-optimal algorithm, extending our design to Markov sources as well as investigating powerful VLEC iterative decoding methods (e.g., cf. [4], [22]) with manageable complexity.

TABLE VI
AVERAGE (AVG) AND MAXIMUM (MAX) NUMBERS OF DECODER BRANCH METRIC COMPUTATIONS FOR THE CODES OF FIGURE 4.

SNR_s	8 dB		9 dB		10 dB		11 dB	
VLEC system	AVG	MAX	AVG	MAX	AVG	MAX	AVG	MAX
Wang's VLEC	3124	11131	3123	4524	3123	4093	3123	4000
Buttigieg's VLEC	3112	16433	3111	5950	3111	4001	3111	4001
Lamy's VLEC	3211	14675	3210	5959	3209	4391	3209	4391
Suboptimal VLEC	3108	10096	3104	4349	3104	3995	3104	3995

TABLE VII
AVERAGE (AVG) AND MAXIMUM (MAX) NUMBERS OF DECODER BRANCH METRIC COMPUTATIONS FOR THE CODES OF FIGURE 5. THE PARAMETER λ USED IN PFSA IS INDICATED INSIDE THE PARENTHESES.

SNR_s		8 dB		9 dB		10 dB		11 dB	
Scheme		AVG	MAX	AVG	MAX	AVG	MAX	AVG	MAX
Code	Decoder								
$(3,1,3)$ TBCC $[54,64,74]$	PFSA(3)	753	2049	739	1518	731	1483	730	1253
$(3,1,4)$ TBCC $[52,66,76]$	PFSA(4)	1466	4192	1444	3298	1435	2916	1432	2528
$(3,1,5)$ TBCC $[47,53,75]$	PFSA(5)	2907	8909	2875	6437	2865	4851	2862	4661
$(3,1,6)$ TBCC $[564,624,754]$	PFSA(6)	5773	21062	5734	12814	5724	9063	5721	8687
Suboptimal VLEC	TPSMAP	2698	8322	2695	7362	2694	3840	2694	3840

REFERENCES

[1] F. Alajaji, N. Phamdo, and T. Fuja, "Channel codes that exploit the residual redundancy in CELP-encoded speech," *IEEE Trans. Speech Audio Process.,* vol. 4, pp. 325–336, Sep. 1996.

[2] E. Ayanoglu and R. Gray, "The design of joint source and channel trellis waveform coders," *IEEE Trans. Inf. Theory*, vol. 33, no. 6, pp. 855–865, Nov. 1987.

[3] V. B. Balakirsky, "Joint source-channel coding with variable length codes," in *Proc. 1997 IEEE Int. Symp. Inform. Theory*, p. 419.

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

[5] M. A. Bernard and B. D. Sharma, "Some combinatorial results on variable length error-correcting codes," *ARS Combinatoria*, vol. 25B, pp. 181–194, 1988.

[6] M. A. Bernard and B. D. Sharma, "A lower bound on average codeword length of variable length error-correcting codes," *IEEE Trans. Inf. Theory*, vol. 36, no. 6, pp. 1474–1475, Nov. 1990.

[7] V. Buttigieg, "Variable-length error-correcting codes," Ph.D. thesis, Univ. of Manchester, England, 1995.

[8] V. Buttigieg, personal communication, 2012.

[9] V. Buttigieg and P. G. Farrell, "Variable-length error-correcting codes," *IEE Proc. Commun.*, vol. 147, no. 4, pp. 211–215, Aug. 2000.

[10] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. MIT Press, 2001.

[11] A. Diallo, C. Weidmann, and M. Kieffer, "Optimizing the free distance of error-correcting variable-length codes," in *Proc. 2010 IEEE Int. Workshop Multimedia Signal Process.*, pp. 245–250.

[12] A. Diallo, C. Weidmann, and M. Kieffer, "Efficient computation and optimization of the free distance of variable-length finite-state joint source-channel codes," *IEEE Trans. Commun.*, vol. 59, no. 4, pp. 1043–1052, Apr. 2011.

[13] A. Diallo, C. Weidmann, and M. Kieffer, "New free distance bounds and design techniques for joint source-channel variable-length codes," *IEEE Trans. Commun.*, vol. 60, no. 10, pp. 3080–3090, Oct. 2012.

[14] P. Duhamel and M. Kieffer, *Joint Source-Channel Decoding: A Cross-Layer Perspective with Applications in Video Broadcasting over Mobile and Wireless Networks.* Academic Press, 2010.

[15] 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.

[16] Y. S. Han, T.-Y. Wu, H.-T. Pai, P.-N. Chen, and S.-L. Shieh, "Priority-first search decoding for convolutional tail-biting codes," in *Proc. 2008 Int. Symp. Inform. Theory and its Applications*, pp. 1–6.

[17] W. E. Hartnett, *Foundation of Coding Theory*. D. Reidel Publishing Co., 1974.

[18] H. Hijazi, A. Diallo, M. Kieffer, L. Liberti, and C. Weidmann, "A MILP approach for designing robust variable-length codes based on exact free distance computations," in *Proc. 2012 Data Compression Conf.*, pp. 257–266.

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

[20] 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.

[21] S. Kaiser and M. Bystrom, "Soft decoding of variable-length codes," in *Proc. 2000 IEEE Int. Conf. on Commun.*, vol. 3, pp. 1203–1207.

[22] J. Kliewer and R. Thobaden, "Iterative joint source-channel decoding of variable-length codes using residual source redundancy," *IEEE Trans. Wireless Commun.*, vol. 4, no. 3, pp. 919–929, May 2005.

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

[24] C. Lamy and F. X. Bergot, "Lower bounds on the existence of binary error-correcting variable-length codes," in *Proc. 2003 IEEE Inform. Theory Workshop*, pp. 300–303.

[25] J. C. Maxted and J. P. Robinson, "Error recovery for variable length codes," *IEEE Trans. Inf. Theory*, vol. 31, no. 6, pp. 794–801, Nov. 1985.

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

[27] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical J.*, vol. 27, pt. I, pp. 379–423; pt. II, pp. 623–656, 1948.

[28] S.-L. Shieh, P.-N. Chen, Y. S. Han, and T.-Y. Wu, "Early-elimination modification for priority-first search decoding," *IEEE Trans. Commun.*, vol. 58, no. 12, pp. 3459–3469, Dec. 2010.

[29] P. Stahl, J. B. Anderson, and R. Johannesson, "Optimal and near-optimal encoders for short and moderate-length tail-biting trellises," *IEEE Trans. Inf. Theory*, vol. 45, no. 7, pp. 2562–2571, Nov. 1999.

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

[31] T. Wenisch, P. F. Swaszek, and A. K. Uht, "Combined error correcting and compressing codes," in *Proc. 2001 IEEE Int. Symp. Inform. Theory*, pp. 238.

[32] T.-Y. Wu, P.-N. Chen, F. Alajaji, and Y. S. Han, "On the construction and MAP decoding of optimal variable-length error-correcting codes," in *Proc. 2011 IEEE Int. Symp. Inform. Theory*, pp. 2223–2227.

[33] Y. Zhong, F. Alajaji, and L. L. Campbell, "On the joint source-channel coding error exponent for discrete memoryless systems," *IEEE Trans. Inf. Theory*, vol. 52, no. 4, pp. 1450–1468, Apr. 2006.

Ting-Yi Wu was born in Tainan, Taiwan, in 1983. He received the B.Sc. and M.Sc. degrees in Computer Science and Information Engineering from National Chi-Nan University, Nantou, Taiwan, in 2005 and 2007, respectively. From 2007 to 2009, he was a research assistant of the Graduate Institute of Communication Engineering, National Taipei University, Taipei, Taiwan. He is currently pursuing the Ph.D. degree in Institute of Communications Engineering, National Chiao-Tung University, Hsinchu, Taiwan. His current research interests include error-control coding and information theory.

Po-Ning Chen (S'93–M'95–SM'01) was born in Taipei, R.O.C. in 1963. He received the B.S. and M.S. degrees in electrical engineering from the National Tsing-Hua University, Taiwan in 1985 and 1987, respectively, and the Ph.D. degree in electrical engineering from University of Maryland, College Park, in 1994. From 1985 to 1987, he was with Image Processing Laboratory in National Tsing-Hua University, where he worked on the recognition of Chinese characters. During 1989, he was with Star Tech. Inc., where he focused on the development of finger-print recognition systems. After the reception of Ph.D. degree in 1994, he jointed Wan Ta Technology Inc. as a vice general manager, conducting several projects on Point-of-Sale systems. In 1995, he became a research staff in Advanced Technology Center, Computer and Communication Laboratory, Industrial Technology Research Institute in Taiwan, where he led a project on Java-based Network Managements. Since 1996, he has been an Associate Professor in the Department of Communications Engineering at the National Chiao-Tung University, Taiwan, and was promoted to a full professor since 2001. He was elected to be the Chair of the IEEE Communications Society Taipei Chapter in 2006 and 2007, during which the IEEE ComSoc Taipei Chapter won the 2007 IEEE ComSoc Chapter Achievement Awards (CAA) and 2007 IEEE ComSoc Chapter of the Year (CoY). He has served as the chairman of the Department of Communications Engineering, National Chiao-Tung University, during 2007–2009.

Dr. Chen received the annual Research Awards from the National Science Council, Taiwan, R.O.C., five years in a row since 1996. He then received the 2000 Young Scholar Paper Award from Academia Sinica, Taiwan. His Experimental Handouts for the course of Communication Networks Laboratory have been awarded as the Annual Best Teaching Materials for Communications Education by the Ministry of Education, Taiwan, R.O.C., in 1998. He has been selected as the Outstanding Tutor Teacher of the National Chiao-Tung University in 2002. He was also the recipient of the Distinguished Teaching Award from the College of Electrical and Computer Engineering, National Chiao-Tung University, Taiwan, in 2003. His research interests generally lie in information and coding theory, large deviation theory, distributed detection and sensor networks.

Fady Alajaji (S'90–M'94–SM'00) received the B.E. degree with distinction from the American University of Beirut, Lebanon, and the M.Sc. and Ph.D. degrees from the University of Maryland, College Park, all in electrical engineering, in 1988, 1990 and 1994, respectively. He held a postdoctoral appointment in 1994 at the Institute for Systems Research, University of Maryland.

In 1995, he joined the Department of Mathematics and Statistics at Queen's University, Kingston, Ontario, where he is currently a Professor of Mathematics and Engineering. Since 1997, he has also been cross-appointed in the Department of Electrical and Computer Engineering at the same university. From 2003 to 2008, he served as chair of the Queen's Mathematics and Engineering program. His research interests include information theory, digital communications, error control coding, joint source-channel coding and data compression.

Dr. Alajaji currently serves as Area Editor and Editor for Source and Source-Channel Coding for the IEEE TRANSACTIONS ON COMMUNICATIONS. He served as organizer and Technical Program Committee member of several international conferences and workshops. He received the Premier's Research Excellence Award from the Province of Ontario.

Yunghsiang S. Han (S'90–M'93–SM'08–F'11) was born in Taipei, Taiwan, 1962. He received B.Sc. and M.Sc. degrees in electrical engineering from the National Tsing Hua University, Hsinchu, Taiwan, in 1984 and 1986, respectively, and a Ph.D. degree from the School of Computer and Information Science, Syracuse University, Syracuse, NY, in 1993.

He was from 1986 to 1988 a lecturer at Ming-Hsin Engineering College, Hsinchu, Taiwan. He was a teaching assistant from 1989 to 1992, and a research associate in the School of Computer and Information Science, Syracuse University from 1992 to 1993. He was, from 1993 to 1997, an Associate Professor in the Department of Electronic Engineering at Hua Fan College of Humanities and Technology, Taipei Hsien, Taiwan. He was with the Department of Computer Science and Information Engineering at National Chi Nan University, Nantou, Taiwan from 1997 to 2004. He was promoted to Professor in 1998. He was a visiting scholar in the Department of Electrical Engineering at University of Hawaii at Manoa, HI from June to October 2001, the SUPRIA visiting research scholar in the Department of Electrical Engineering and Computer Science and CASE center at Syracuse University, NY from September 2002 to January 2004 and July 2012 to June 2013, and the visiting scholar in the Department of Electrical and Computer Engineering at University of Texas at Austin, TX from August 2008 to June 2009. He was with the Graduate Institute of Communication Engineering at National Taipei University, Taipei, Taiwan from August 2004 to July 2010. From August 2010, he is with the Department of Electrical Engineering at National Taiwan University of Science and Technology as Chair professor. His research interests are in error-control coding, wireless networks, and security.

Dr. Han was a winner of the 1994 Syracuse University Doctoral Prize and a Fellow of IEEE.