An Efficient Tree Search Algorithm for the Free Distance of Variable-Length Error-Correcting Codes

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

***Abstract*— We propose an efficient tree search algorithm for determining the free distance of variable-length error-correcting codes (VLECs). A main idea behind the algorithm is to structure all pairs of code word-concatenated sequences as a tree, in which we seek the pair of sequences that determine the free distance. In order to speed up the algorithm, we establish constraints that do not compromise optimality in determining the free distance. Experimental results on VLECs algorithmically constructed for the English alphabet show that our algorithm requires a considerably smaller number of bitwise distance computations and covers a much smaller number of tree nodes than Dijkstra's algorithm operating over the pairwise distance graph while being a hundred times faster in terms of execution time.**

***Index Terms*— Joint source-channel coding, variable length error-correcting codes, free distance, Dijkstra's algorithm.**

I. Introduction

THE source-channel separation principle states that the source and channel coding operations can be separately designed and applied in tandem without loosing system optimality in terms of sending a source over a noisy point-to-point channel and reliably recovering it [1]. This result, however, relies on the key assumption that unlimited complexity and coding delay can be tolerated by the system, which is unrealistic in practical communication systems.

Many works have demonstrated that joint source-channel coding (JSCC) can considerably outperform separate source-channel coding (e.g., see [2]–[5] and the references therein and thereafter), particularly when the system has severe complexity and delay constraints. JSCC systems are usually constructed by either coordinating the source and channel coding operations or combining them in a single operation. In this letter, we consider JSCC designs of the latter type using variable-length error-correcting codes (VLECs).

In one of the original works on VLEC design, Buttigieg [6], [7] showed that the *free distance* of a VLEC can be utilized to predict its error performance. For a given VLEC codebook $\mathcal{C}$, the free distance is defined as

$$d_{\text{free}}(\mathcal{C}) \triangleq \min_{N\geq 1}\{d(\boldsymbol{a},\boldsymbol{b}) : \boldsymbol{a},\boldsymbol{b} \in \mathcal{X}_N(\mathcal{C}) \text{ and } \boldsymbol{a}\neq\boldsymbol{b}\},$$

where $d(\boldsymbol{a},\boldsymbol{b})$ is the Hamming distance between two binary bitstreams $\boldsymbol{a}$ and $\boldsymbol{b}$, and

$$\mathcal{X}_N(\mathcal{C}) \triangleq \bigcup_{L=1}^{N}\left\{\boldsymbol{c}_1\boldsymbol{c}_2\cdots\boldsymbol{c}_L : \boldsymbol{c}_i\in\mathcal{C} \text{ and } \sum_{i=1}^{L}|\boldsymbol{c}_i| = N\right\}$$

is the set of codeword-concatenated sequences of length N. Here, $|\boldsymbol{c}_i|$ denotes the length of bitstream $\boldsymbol{c}_i$. Buttigieg's finding simplifies the algorithmic strategy of searching for good VLEC designs as to either *fix a free distance lower bound and minimize the average codeword length* [8], or *fix a set of codeword lengths and maximize the free distance* [9]–[11]. The algorithmic efficiency of either strategies relies heavily on how effectively the free distance of candidate VLECs can be determined.

In [8], $d_{\text{free}}(\mathcal{C})$ is calculated using the trellis diagram suggested in [6]. A sufficient condition based on converge and diverge distances is applied to quickly exclude those VLEC designs that violate a free distance lower bound. A different approach was used in [9], where the pairwise distance graph (PDG) of the VLEC is first generated and then Dijkstra's algorithm is applied to the PDG to determine the free distance.

In this work, a new tree search algorithm for determining $d_{\text{free}}(\mathcal{C})$ is proposed. The algorithm, which is a refinement of the one introduced in [6, Sec. 3.5.1.1] for a similar task, is shown to be much more efficient than the PDG search in [9]. Different from the PDG search, we structure all pairs of codeword-concatenated sequences as a tree, over which we locate the pair of sequences that result in $d_{\text{free}}(\mathcal{C})$. In order to speed up the search process, we introduce constraints that yield no loss of optimality in determining $d_{\text{free}}(\mathcal{C})$. Experiments executed for VLECs designed for the English alphabet demonstrate the efficiency of the proposed algorithm.

II. Determination of the Free Distance

A key problem for the determination of $d_{\text{free}}(\mathcal{C})$ is to have an algorithmically tractable structure, over which all pairs of equal-length codeword-concatenated sequences can be either exhausted, or excluded if their pairwise distance exceeds $d_{\text{free}}(\mathcal{C})$. Different from the PDG search adopted in [9], we list all pairs of codeword-concatenated sequences in a tree. The construction of the tree begins with the generation of $\binom{|\mathcal{C}|}{2} = \frac{|\mathcal{C}|\,(|\mathcal{C}|-1)}{2}$ child nodes from the dummy root node, each of which contains a pair of distinct codewords from $\mathcal{C}$ as

Manuscript received October 16, 2017; revised November 17, 2017; accepted November 20, 2017. Date of publication November 24, 2017; date of current version March 8, 2018. This work was supported in part by the Minister of Science and Technology of Taiwan under Grants MOST 105-2917-I-564-048, MOST 103-2221-E-009-044-MY2 and MOST 105-2221-E-009-009-MY3 and by the National Natural Science Foundation of China (Grant No. 61671007). The associate editor coordinating the review of this letter and approving it for publication was C. Feng. *(Corresponding author: Ting-Yi Wu.)*

C. Huang and P.-N. Chen are with the Institute of Communication Engineering, National Chiao Tung University, Hsinchu 300, Taiwan.

T.-Y. Wu is with the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Champaign, IL 61801 USA (e-mail: mavericktywu@gmail.com).

F. Alajaji is with the Department of Mathematics and Statistics, Queen's University, Kingston, ON K7L 3N6, Canada.

Y. S. Han is with the School of Electrical Engineering and Intelligentization, Dongguan University of Technology, Dongguan 523808, China.

Digital Object Identifier 10.1109/LCOMM.2017.2777441

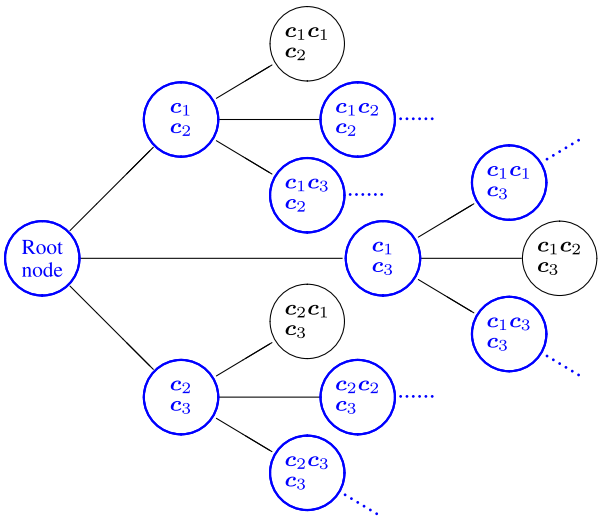


Fig. 1. Example of a search tree for $\mathcal{C} = \{\boldsymbol{c}_1 = 0, \boldsymbol{c}_2 = 10, \boldsymbol{c}_3 = 111\}$. The nodes in black circles contain two equal-length sequences and hence have no child nodes.

illustrated in Fig. 1 for $\mathcal{C} = \{\boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{c}_3\}$. The three child nodes extended from the root node in Fig. 1 thus contain $(\boldsymbol{c}_1, \boldsymbol{c}_2)$, $(\boldsymbol{c}_2, \boldsymbol{c}_3)$ and $(\boldsymbol{c}_1, \boldsymbol{c}_3)$, respectively.

Next, from each child node, we extend the sequence with a shorter length by concatenating it with a codeword. Thus, every child node has $|\mathcal{C}|$ extended nodes. For example, since $|\mathcal{C}| = 3$ in Fig. 1, each child node has three extended nodes.

When the lengths of the two sequences contained in a node are equal, no further extension is performed. By this rule, the nodes in black circles in Fig. 1 are no longer extended, and hence have no child nodes.

The first property we derive to help simplify the tree search is that it suffices to consider *indecomposable* pairs of equal-length sequence.

Definition 1(Indecomposable Equal-Length Sequence Pair): A pair of equal-length sequences

$$\boldsymbol{a} = \boldsymbol{c}_{i_1}\boldsymbol{c}_{i_2}\cdots\boldsymbol{c}_{i_m} \quad \text{and} \quad \boldsymbol{b} = \boldsymbol{c}_{j_1}\boldsymbol{c}_{j_2}\cdots\boldsymbol{c}_{j_n}$$

is *indecomposable* if

$$\sum_{k=1}^{m'-1} |\boldsymbol{c}_{i_k}| \neq \sum_{k=1}^{n'-1} |\boldsymbol{c}_{j_k}|$$

for every $1 < m' \leq m$ and $1 < n' \leq n$.

Lemma 1: When computing $d_{\text{free}}(\mathcal{C})$, it suffices to consider only indecomposable equal-length sequence pairs.

Proof: If $(\boldsymbol{a}, \boldsymbol{b}) = \left(\boldsymbol{c}_{i_1}\boldsymbol{c}_{i_2}\cdots\boldsymbol{c}_{i_m}, \boldsymbol{c}_{j_1}\boldsymbol{c}_{j_2}\cdots\boldsymbol{c}_{j_n}\right)$ is an equal-length sequence pair but not indecomposable, there must exist m' and n' with $1 < m' \leq m$ and $1 < n' \leq n$ such that $\sum_{k=1}^{m'-1} |\boldsymbol{c}_{i_k}| = \sum_{k=1}^{n'-1} |\boldsymbol{c}_{j_k}|$. Since

$$\begin{aligned} &d\left(\boldsymbol{c}_{i_1}\boldsymbol{c}_{i_2}\cdots\boldsymbol{c}_{i_{m'-1}}, \boldsymbol{c}_{j_1}\boldsymbol{c}_{j_2}\cdots\boldsymbol{c}_{j_{n'-1}}\right) \\ &\quad \leq d\left(\boldsymbol{c}_{i_1}\boldsymbol{c}_{i_2}\cdots\boldsymbol{c}_{i_{m'-1}}, \boldsymbol{c}_{j_1}\boldsymbol{c}_{j_2}\cdots\boldsymbol{c}_{j_{n'-1}}\right) \\ &\qquad\qquad + d\left(\boldsymbol{c}_{i_{m'}}\cdots\boldsymbol{c}_{i_m}, \boldsymbol{c}_{j_{n'}}\cdots\boldsymbol{c}_{j_n}\right) \\ &\quad = d\left(\boldsymbol{c}_{i_1}\boldsymbol{c}_{i_2}\cdots\boldsymbol{c}_{i_m}, \boldsymbol{c}_{j_1}\boldsymbol{c}_{j_2}\cdots\boldsymbol{c}_{j_n}\right) = d\left(\boldsymbol{a}, \boldsymbol{b}\right), \end{aligned}$$

excluding $(\boldsymbol{a}, \boldsymbol{b})$ in the tree search will not compromise the determination of $d_{\text{free}}(\mathcal{C})$. The lemma therefore holds. ■

It could happen that the tree is extended indefinitely. For example, if $\sum_{k=1}^{m'-1} |\boldsymbol{c}_{i_k}| \neq \sum_{k=1}^{n'-1} |\boldsymbol{c}_{j_k}|$ for every $1 < m' \leq m$ and $1 < n' \leq n$, but $\sum_{k=m''}^{m} |\boldsymbol{c}_{i_k}| = \sum_{k=n''}^{n} |\boldsymbol{c}_{j_k}|$ for some $1 < m'' \leq m$ and $1 < n'' \leq n$, an indefinite extension occurs because no ancestor nodes of $(\boldsymbol{a}', \boldsymbol{b}')$ as defined below contain equal-length sequences and hence their extensions will never be terminated:

$$\begin{cases} \boldsymbol{a}' = \boldsymbol{c}_{i_1}\cdots\boldsymbol{c}_{i_{m''-1}} \underbrace{\boldsymbol{c}_{i_{m''}}\cdots\boldsymbol{c}_{i_m}}_{\text{periodic pattern}} \cdots \underbrace{\boldsymbol{c}_{i_{m''}}\cdots\boldsymbol{c}_{i_m}}_{\text{periodic pattern}}; \\ \boldsymbol{b}' = \boldsymbol{c}_{j_1}\cdots\boldsymbol{c}_{j_{n''-1}} \underbrace{\boldsymbol{c}_{j_{n''}}\cdots\boldsymbol{c}_{j_n}}_{\text{periodic pattern}} \cdots \underbrace{\boldsymbol{c}_{j_{n''}}\cdots\boldsymbol{c}_{j_n}}_{\text{periodic pattern}}. \end{cases} \tag{1}$$

In order to mitigate such an indefinite node extension problem, a criterion based on the converge and diverge distances [6] is provided and proven in the next lemma.

Definition 2 (Converge and Diverge Distances): For a sequence pair $(\boldsymbol{a}, \boldsymbol{b})$ with $|\boldsymbol{a}| \leq |\boldsymbol{b}|$, the converge distance $d_{\text{C}}(\boldsymbol{a}, \boldsymbol{b})$ and the diverge distance $d_{\text{D}}(\boldsymbol{a}, \boldsymbol{b})$ are defined as

$$d_{\text{C}}(\boldsymbol{a}, \boldsymbol{b}) \triangleq d(\boldsymbol{a}, \boldsymbol{b}_{\text{suff}}) \quad \text{and} \quad d_{\text{D}}(\boldsymbol{a}, \boldsymbol{b}) \triangleq d(\boldsymbol{a}, \boldsymbol{b}_{\text{pref}}),$$

where $\boldsymbol{b}_{\text{suff}}$ and $\boldsymbol{b}_{\text{pref}}$ are the suffix and prefix of $\boldsymbol{b}$, respectively, satisfying $|\boldsymbol{b}_{\text{suff}}| = |\boldsymbol{b}_{\text{pref}}| = |\boldsymbol{a}|$.[1]

Two notations will be used. Let $D_{\text{min-C}}$ be the minimum converge distance among all distinct codeword pairs of unequal length, i.e.,

$$D_{\text{min-C}} \triangleq \min_{\boldsymbol{c}_{i_1}, \boldsymbol{c}_{i_2} \in \mathcal{C} : |\boldsymbol{c}_{i_1}| < |\boldsymbol{c}_{i_2}|} d_{\text{C}}(\boldsymbol{c}_{i_1}, \boldsymbol{c}_{i_2}).$$

Denote by tmp-d_{free} the smallest distance among all nodes that have thus far been visited by a tree search process and that contain an equal-length sequence pair.

Lemma 2: If the sequence pair $(\boldsymbol{a}, \boldsymbol{b})$ contained in a node satisfies

$$d_{\text{D}}(\boldsymbol{a}, \boldsymbol{b}) + D_{\text{min-C}} \geq \texttt{tmp-}d_{\text{free}}, \tag{2}$$

then the removal of this node over the search tree will never compromise the determination of $d_{\text{free}}(\mathcal{C})$.

Proof: Without loss of generality, we assume that at least one equal-length sequence pair has been visited by the tree search process such that tmp-d_{free} is finite.

Since an offspring node $(\boldsymbol{a}', \boldsymbol{b}')$ of node $(\boldsymbol{a}, \boldsymbol{b})$ satisfies

$$d(\boldsymbol{a}', \boldsymbol{b}') \geq d_{\text{D}}(\boldsymbol{a}, \boldsymbol{b}) + D_{\text{min-C}},$$

provided $|\boldsymbol{a}'| = |\boldsymbol{b}'|$, and since the search process has already identified a pair of candidate sequences whose pairwise distance is equal to tmp-d_{free}, we obtain from (2) that

$$d(\boldsymbol{a}', \boldsymbol{b}') \geq \texttt{tmp-}d_{\text{free}} \geq d_{\text{free}}(\mathcal{C}).$$

Consequently, further extension of node $(\boldsymbol{a}, \boldsymbol{b})$ will never yield a pair of equal-length sequences with their pairwise distance less than tmp-d_{free}. Its removal, therefore, will not compromise the determination of $d_{\text{free}}(\mathcal{C})$. ■

It is straightforward from the definition of $d_{\text{free}}(\mathcal{C})$ that a node labeled with sequence pair $(\boldsymbol{a}, \boldsymbol{b})$ shall be removed during

[1]For clarity, in this letter, $\boldsymbol{b}$ and $\boldsymbol{b}'$ (resp. $\boldsymbol{a}$ and $\boldsymbol{a}'$) are used to denote sequences of codewords, whereas $\boldsymbol{b}_{\text{suff}}$, $\boldsymbol{b}_{\text{pref}}$ and $\boldsymbol{b}_{\text{last}}$ (cf. Lemma 3) denote binary sequences of arbitrary length.

the algorithmic search of $d_{\text{free}}(\mathcal{C})$ if $d_{\text{D}}(\boldsymbol{a},\boldsymbol{b})$ exceeds the recorded threshold value $\texttt{tmp}$-d_{free}. The above lemma consequentially provides an early elimination of nodes by lowering the threshold by the amount of $D_{\min\text{-C}}$. Since the number of nodes increases exponentially as nodes are expanded further, early elimination of nodes even with a small positive $D_{\min\text{-C}}$ can speed up considerably the search process; this will be confirmed by our experimental results.

In general, the diverge distance is non-decreasing along a path of the tree to be constructed, and hence (2) is expected to be ultimately violated. However, in an unusual situation, it may happen that the two sequences $(\boldsymbol{a}',\boldsymbol{b}')$ defined in (1) have the same diverge distance as $\boldsymbol{c}_{i_1}\cdots\boldsymbol{c}_{i_{m''-1}}$ and $\boldsymbol{c}_{j_1}\cdots\boldsymbol{c}_{j_{n''-1}}$, regardless of the number of the periodic patterns. As such, (2) is always valid, and fails to stop the indefinite extension. This can be resolved by the next lemma.

Lemma 3: For a node that contains sequence pair $(\boldsymbol{a},\boldsymbol{b})$ with $|\boldsymbol{a}|<|\boldsymbol{b}|$, let $\boldsymbol{b}=\boldsymbol{b}_{\text{pref}}\boldsymbol{b}_{\text{last}}$ with $|\boldsymbol{b}_{\text{last}}|=|\boldsymbol{b}|-|\boldsymbol{a}|$. If two or more nodes have the same $\boldsymbol{b}_{\text{last}}$, then retaining one with the minimum diverge distance among them (and eliminating all others) will not compromise the determination of $d_{\text{free}}(\mathcal{C})$.

Proof: Two or more nodes having identical $\boldsymbol{b}_{\text{last}}$ must have identical structure for their offspring nodes. Since the diverge distance patterns of the offspring nodes only depend on $\boldsymbol{b}_{\text{last}}$, it suffices to keep one node with the minimum diverge distance. ∎

A particular case of Lemma 3 that may help figure out how an indefinite extension could occur and hence avoid it in VLEC code designs is that $\boldsymbol{c}_{i_{m''}}\cdots\boldsymbol{c}_{i_m}$ in (1) is the $(n''-m'')$-bit right circular shift of $\boldsymbol{c}_{j_{m''}}\cdots\boldsymbol{c}_{j_m}$ Then, irrespective of the number of periodic patterns in (1), $d_{\text{D}}(\boldsymbol{a}',\boldsymbol{b}')$ remains the same. Such an indefinite extension can therefore be avoided by adopting the speedup technique from Lemma 3 in the algorithmic search of $d_{\text{free}}(\mathcal{C})$.

We summarize the proposed tree search algorithm for finding $d_{\text{free}}(\mathcal{C})$ as follows.

Tree Search Algorithm: Each node is labeled with two sequences $(\boldsymbol{a},\boldsymbol{b})$.

Step 1. Initialization. *Generates* $\binom{|\mathcal{C}|}{2}$ *child nodes from the root node. Compute the minimum converge distance* $D_{\min\text{-}\mathcal{C}}$. *Set* $\texttt{tmp}$-$d_{\text{free}}=\infty$.

Step 2. $\texttt{tmp}$-d_{free} Update and Equal-length Check. *For every node* $(\boldsymbol{a},\boldsymbol{b})$, *if* $|\boldsymbol{a}|=|\boldsymbol{b}|$ *and* $d(\boldsymbol{a},\boldsymbol{b})<\texttt{tmp}$-$d_{\text{free}}$, *update* $\texttt{tmp}$-$d_{\text{free}}=d(\boldsymbol{a},\boldsymbol{b})$.
Remove all nodes with an equal-length sequence pair.

Step 3. Early Elimination. *Remove all nodes satisfying* (2).

Step 4. $\boldsymbol{b}_{last}$-Check. *Let* $\boldsymbol{b}=\boldsymbol{b}_{\text{pref}}\boldsymbol{b}_{\text{last}}$ *with* $|\boldsymbol{b}_{last}|=|\boldsymbol{b}|-|\boldsymbol{a}|$, *where* $\boldsymbol{b}$ *denotes the longer one of the two sequences a node contains. If two or more nodes have the same* $\boldsymbol{b}_{last}$, *retain one with the minimum diverge distance and remove all others.*

Step 5. End-of-Search Check and Node Extension. *If there are no nodes remaining, set* $d_{\text{free}}(\mathcal{C})=\texttt{tmp}$-$d_{\text{free}}$ *and stop the algorithm; else, among all remaining nodes, set the node with the minimum diverge distance as the parent node, and generate* $|\mathcal{C}|$ *child nodes by concatenating the shorter sequence with a codeword, and remove the parent node of these* $|\mathcal{C}|$ *child nodes.*[2] *Go to Step 2.*

III. Experimental Results

In this section, the efficiency of the proposed tree search algorithm is examined and compared with that of the PDG search algorithm in [9]. The experiments were programmed using the *C language* under a 64-*bit Linux* operating system *(Ubuntu 16.04.3 LTS)* executed over an *Intel server system* with 2 *Xeon(R) E5-2620 v3 2.40GHz CPUs* and *64GB* memory. Two main factors that may affect the efficiency of an algorithm are computational complexity and memory access burden. Irrespective of programming skills, the former can be assessed via the operation that repeats mostly for a $d_{\text{free}}(\mathcal{C})$-determination algorithm, i.e., the bit-wise XOR (i.e., bit-wise computation of Hamming distance), while the latter is proportional to the number of nodes required to be extended. Based on our implementations, the execution times of both algorithms are also reported. The first eight VLECs to be studied are from [8, Table II and Table III], which are algorithmically constructed for the 26-letter English alphabet subject to free distance bounds of 3, 5, 7 and 9 in combination with two source distributions. A comparison is also conducted based on the VLEC from [12] (which is a follow-up work of [9]).

It can be observed from Table I that the number of bit-wise XOR operations required by the proposed tree search algorithm is considerably smaller than the number of bit-wise XOR operations required by the PDG search algorithm. It should be mentioned that the PDG search algorithm implemented here is not the original one in [9], where the search of $d_{\text{free}}(\mathcal{C})$ is conducted after the construction of the entire PDG, but an improved version of it, where nodes over the PDG are extended only when the computation of the respective bit-wise distance is required. We then compared the number of nodes required by each algorithm, and found that the superiority of the proposed tree search over the PDG search is mainly due to the effectiveness of node removal through Steps 3 and 4. This observation can be more easily seen by highlighting those edges over the PDG that are "equivalently" XOR-operated by the proposed tree search. Based on the PDG for VLEC $\mathcal{C}=\{\boldsymbol{c}_1=00,\boldsymbol{c}_2=011,\boldsymbol{c}_3=1001\}$, Dijkstra's algorithm is adopted to determine $d_{\text{free}}(\mathcal{C})$ in [9], where only 22 XOR operations are needed as shown in Fig. 2(a). The proposed tree search algorithm reduces the XOR operations down to 16 as shown in Fig. 2(b), where seven XOR operations are saved owing to (2) but one XOR operation is added because our distance computation is codeword-based and hence the bit-wise XORs corresponding to a codeword must be all included. This reduction of XOR operations is reflected by the reduction of node expansions (e.g., from 19 visited nodes down to 15 visited nodes in Fig. 2). Note that it is conceptually hard to compare directly the number of nodes required by both algorithms since they operate on different search structures.

[2]Notably, the algorithm essentially keeps a list of unexplored nodes, including the $|\mathcal{C}|$ child nodes just generated. The algorithm stops when these unexplored nodes are all removed via the node removal in Steps 2-4.

TABLE I

NUMBERS OF NODES, BIT-WISE XOR OPERATIONS AND EXECUTION TIMES FOR THE PROPOSED TREE SEARCH ALGORITHM AND THE PDG SEARCH ALGORITHM. IN ADDITION TO THE NUMBER OF XORS REQUIRED FOR THE EXECUTION OF THE ENTIRE ALGORITHM, THE NUMBER OF XORS REQUIRED TO COMPUTE $D_{\min\text{-C}}$ IS SEPARATELY LISTED IN PARENTHESES. THE EXECUTION TIME REPORTED IS THE TOTAL TIME OF REPEATING AN ALGORITHM 1000 TIMES. THE RATIO IN THE LAST COLUMN IS EQUAL TO (EXECUTION TIME/NUMBER OFNODES) $\times 10^4$

VLECs from [8, Table II]

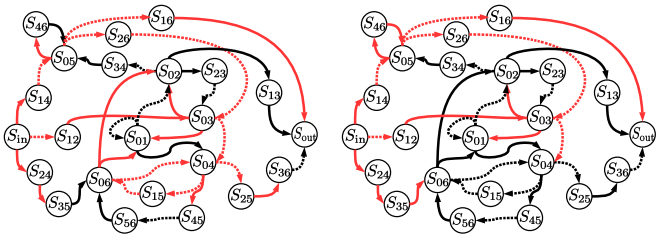
	Nodes	XORs	Execution time	Ratio
Tree ($d^*_{\text{free}} = 3$)	33	3991 (1828)	0.028 secs	8.48
PDG ($d^*_{\text{free}} = 3$)	4468	8840	3.085 secs	6.90
Tree ($d^*_{\text{free}} = 5$)	56	5731 (2760)	0.034 secs	6.07
PDG ($d^*_{\text{free}} = 5$)	16852	23487	10.703 secs	6.35
Tree ($d^*_{\text{free}} = 7$)	33	6527 (2995)	0.036 secs	10.91
PDG ($d^*_{\text{free}} = 7$)	26580	33242	17.591 secs	6.62
Tree ($d^*_{\text{free}} = 9$)	99	8492 (4090)	0.042 secs	4.24
PDG ($d^*_{\text{free}} = 9$)	48396	58026	31.910 secs	6.59

VLECs from [8, Table III]

	Nodes	XORs	Execution time	Ratio
Tree ($d^*_{\text{free}} = 3$)	32	4005 (1855)	0.028 secs	8.75
PDG ($d^*_{\text{free}} = 3$)	4839	9183	3.229 secs	6.67
Tree ($d^*_{\text{free}} = 5$)	14	5413 (2561)	0.033 secs	23.57
PDG ($d^*_{\text{free}} = 5$)	19769	25791	13.002 secs	6.58
Tree ($d^*_{\text{free}} = 7$)	47	6758 (3209)	0.038 secs	8.09
PDG ($d^*_{\text{free}} = 7$)	33686	41334	26.648 secs	7.91
Tree ($d^*_{\text{free}} = 9$)	91	8453 (4071)	0.043 secs	4.73
PDG ($d^*_{\text{free}} = 9$)	52851	62395	33.181 secs	6.28

VLEC from [12]

	Nodes	XORs	Execution time	Ratio
Tree ($d^*_{\text{free}} = 7$)	62	6281 (2814)	0.036 secs	5.81
PDG ($d^*_{\text{free}} = 7$)	30221	37733	18.537 secs	6.13



(a) XORs by Dijkstra's algorithm (b) XORs by the proposed algorithm

Fig. 2. PDG for $\mathcal{C} = \{\boldsymbol{c}_1 = 00, \boldsymbol{c}_2 = 011, \boldsymbol{c}_3 = 1001\}$. The solid arrows and the dotted arrows correspond to the PDG edge labels of 1 and 0, respectively. The bit-wise XOR operations taken by an algorithm is highlighted by color red.

The interpretation effort made in Fig. 2 is to figure visually what would be required when operating the proposed tree search algorithm over the PDG.

The ratios between execution time and numbers of nodes are also calculated in Table I. These ratios lie between 4.24 and 10.91 except for one, indicating that the execution efficiency of our implementations are approximately proportional to the number of node extensions. The result verifies that the superiority of the proposed tree search over the PDG search is mainly due to the effective node removal through Steps 3 and 4.

We close the section by remarking that in addition to the overall number of XORs required for the execution of the entire algorithm, the number of XORs required to compute $D_{\min\text{-C}}$ is separately listed in parentheses in Table I. This indicates that a large portion of XOR operations is actually spent on the computation of $D_{\min\text{-C}}$.

IV. CONCLUSION

In this letter, a new tree search algorithm for the determination of the free distance of a VLEC was proposed. By incorporating constraints to effectively prune the search tree, the algorithm was shown to be much more efficient than state-of-the-art techniques such as applying Dijkstra's algorithm over the PDG. Future work includes improving the efficiency of the algorithmic VLEC construction, e.g., in [8], and designing better codes than existing ones. Another worthwhile direction is to apply similar node removal constraints to the PDG search to further improve its efficiency.

ACKNOWLEDGMENT

We sincerely thank Diallo *et al.* [9] for providing their code construction program, including the PDG search routine, as a valuable reference for our study. We are also grateful to the anonymous reviewers for their pertinent and constructive comments which were instrumental in the derivation of Lemma 3, the amelioration of the tree search algorithm and a better presentation of the experimental results.

REFERENCES

[1] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, pp. 379–493, Mar. 1948.
[2] F. I. Alajaji, N. C. Phamdo, and T. E. Fuja, "Channel codes that exploit the residual redundancy in CELP-encoded speech," *IEEE Trans. Speech Audio Process.*, vol. 4, no. 5, pp. 325–336, Sep. 1996.
[3] E. Ayanoglu and R. Gray, "The design of joint source and channel trellis waveform coders," *IEEE Trans. Inf. Theory*, vol. IT-33, no. 6, pp. 855–865, Nov. 1987.
[4] P. Duhamel and M. Kieffer, *Joint Source-Channel Decoding: A Cross-Layer Perspective with Application in Video Broadcasting Over Mobil and Wireless Networks*. San Francisco, CA, USA: Academic, 2010.
[5] 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.
[6] V. Buttigieg, "Variable-length error-correcting codes," Ph.D. dissertation, Dept. Elect. Eng., Univ. Manchester, Manchester, U.K., 1995.
[7] V. Buttigieg and P. G. Farrell, "Variable-length error-correcting codes," *IEE Proc.-Commun.*, vol. 147, no. 4, pp. 211–215, Aug. 2000.
[8] T.-Y. Wu, P.-N. Chen, F. Alajaji, and Y. S. Han, "On the design of variable-length error-correcting codes," *IEEE Trans. Commun.*, vol. 61, no. 9, pp. 3553–3565, Sep. 2013.
[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 Process. (MMSP)*, Oct. 2010, pp. 245–250.
[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," *IEEE Trans. Commun.*, vol. 59, no. 4, pp. 1043–1052, Apr. 2011.
[11] 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.
[12] 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 computation," in *Proc. Data Compression Conf. (DCC)*, Apr. 2012, pp. 257–266.