Analysis and Practice of Uniquely Decodable One-to-One Code

Chin-Fu Liu, Hsiao-feng (Francis) Lu, and Po-Ning Chen
Dept. of Electrical & Computer Engineering
National Chiao Tung University, Taiwan 300, ROC
Email: hubert.liu.1031@gmail.com, francis@mail.nctu.edu.tw, poning@faculty.nctu.edu.tw

***Abstract*—In this paper, we consider the uniquely decodable one-to-one code (UDOOC) that is obtained by inserting a comma indicator, termed the *unique word* (UW), between consecutive one-to-one codewords for separation. As such, we analyze a class of UDOOCs and present practical algorithms for encoding and decoding such codes. Specifically, for various cases of UWs, we investigate the number of length-n codewords of UDOOCs and their asymptotic growth rates in n. The proposed encoding and decoding algorithms of UDOOCs can be implemented in parallel at low computational complexity without storing the codebook. Simulation results show that for proper choices of UWs, UDOOCs can achieve better compression efficiency than Lempel-Ziv codes even when the source is not statistically independent.**

I. Introduction and Motivation

In the literature, there are basically two categories of source coding, one for the compression of a sequence of source letters and the other, for a single "one shot" source symbol. A famous representative for the former is the Huffman code, while the so-called one-to-one code (OOC) prevails the latter.

Huffman code yields the minimum average codeword length under the premise of unique decodability, a property required for the compression of a sequence of source letters. Although optimal in principle, it may encounter several obstacles in implementation. For example, the rare codewords are exceedingly long in length, thereby hampering the efficience of decoding. Other practical obstacles include 1) the codebook may need to be priori stored for encoding and decoding, which is impractical for source with moderately large alphabet size, 2) the decoding must be sequential, not in parallel, and 3) error propagation may occur.

In contrast to the Huffman code, the OOC, when being exempted from the constraint of unique decodability, can achieve a smaller average codeword length and hence requires a codebook comprised of codewords of much shorter length.

In this work, we intend to retain the use of OOCs and propose a scheme that can be processed in parallel for the compression of a sequence of source letters. Specifically, we introduce a *unique word* (UW), behaving like a "comma" indicator, to separate the successive OOC codewords. We further prohibit the UW from appearing as an internal subword[1] of the concatenation of a UW, an OOC codeword and another repeated UW in sequence. Any word that does not violate the above prohibition is a codeword. This results in a uniquely decodable OOC (abbreviated as UDOOC). The UDOOC offers many advantages in its practice and requires only a small overhead for UW. For example, the decoder can first locate all the UWs in a bit stream, parsing it into separate codewords, which allows for a subsequent parallel decoding with much shorter latency. The use of UW also limits the extent of error propagation.

We would like to stress that while the use of UW might not be new, UDOOC remains as a novel concept. It has a distinct requirement that the UW cannot appear internally in a stream consisting of a codeword prefixed and suffixed by the same UW. This is the key to the fast identification of codewords for parallel decoding. It is also possible to place two UWs side by side with nothing in between to produce a *null* codeword. We now give a formal definition of UDOOC.

Definition 1: Given UW $\boldsymbol{k} = k_1 k_2 \dots k_L \in \mathbb{F}^L$ of length L, where $\mathbb{F} = \{0, 1\}$ is the alphabet of code symbol, we say $\mathcal{C}_{\boldsymbol{k}}(n)$ is a UDOOC of length n associated with $\boldsymbol{k}$ if it contains all length-n tuples $\boldsymbol{b} = b_1 \dots b_n$ such that $\boldsymbol{k}$ is not an internal subword of the concatenated word $\boldsymbol{kbk}$. In particular, we set $\mathcal{C}_{\boldsymbol{k}}(0) := \emptyset$, the empty set, corresponding to the null word. The overall UDOOC associated with $\boldsymbol{k}$, denoted by $\mathcal{C}_{\boldsymbol{k}}$, is given by

$$\mathcal{C}_{\boldsymbol{k}} = \bigcup_{n \geq 0} \mathcal{C}_{\boldsymbol{k}}(n).$$

Before giving more details, below we provide an example to illustrate how to generate a UDOOC and how it is used in encoding and decoding.

Example 1: Let $\boldsymbol{k} = 00$ be the UW. To prohibit codewords as well as the concatenation of UW and codeword from containing 00 as an internal subword, the following necessary and sufficient constraints are required for codeword $\boldsymbol{c} = c_1 \dots c_n$.

- 00 cannot be a subword of $\boldsymbol{c}$. This means that within $\boldsymbol{c}$,
  - (C1): "0" can only be followed by "1".
  - (C2): "1" can be followed by either "0" or "1".
- 00 cannot be an internal subword of $\boldsymbol{kc}$ or $\boldsymbol{ck}$. This implies
  - (C3): $c_1 = c_n = 1$.

Following constraints (C1)-(C3), we can place the possible codewords on a code tree as exemplified in Fig. 1. In the code tree, each path from root node "null" to a gray-shaded node represents a codeword. Thus, the legitimate codewords for UW "00" include "null", 1, 11, 101, 111, 1011, 1101,

[1] We say $\boldsymbol{a} = a_1 \dots a_m$ is not an internal subword of $\boldsymbol{b} = b_1 \dots b_n$ if there does not exist i such that $b_i \dots b_{i+m-1} = \boldsymbol{a}$ for all $1 < i < n-m+1$.

978-1-4799-0446-4/13/$31.00 ©2013 IEEE

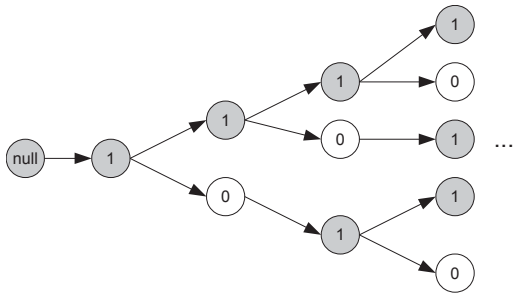

Fig. 1. The code tree for UW "00".

and 1111, etc. Notably, we only show the codewords of length up to 4, while the tree can grow indefinitely in depth. At the decoding stage, assume the received bit-stream is 00100001100101001111100. Knowing that 00 is the UW, the decoder first locates all 00s in the bit stream and interprets the stream as ",1,null,11,101,1111," where UWs are replaced by commas. Thus this stream is separated into five codewords and parallel decoding of them can then be conducted.

A natural question that follows is how many codewords are there for a specific UW. This quantity may be of research and practical interest for UDOOC to be useful. To calculate the number of codewords of length n, let a_n and b_n be respectively the number of "1"-nodes and the number of "0"-nodes at the nth level of the code tree. From conditions (C1) and (C2), we have the following recursions,

$$a_n = a_{n-1} + b_{n-1}, \qquad b_n = a_{n-1}.$$

This yields the renowned Fibonacci sequence for a_n, i.e., $a_n = a_{n-1} + a_{n-2}$. Constraint (C3) shows that only the "1"-nodes can be gray-shaded and result in a codeword. Thus, the number of codewords of length n is a_n. ■

The rest of the paper is organized as follows. Section II investigates the number of length-n codewords of UDOOC and its asymptotic growth rate. Section III gives the efficient encoding and decoding algorithms for UDOOCs with two specific UWs. They can be implemented in parallel and without the need of pre-storing the codebook. Section IV presents simulations of the compression performance of UDOOCs.

II. Asymptotic Growth Rate of UDOOCs

To construct a UDOOC, the UW must be first identified. Let $c_{\boldsymbol{k},n} = |\mathcal{C}_{\boldsymbol{k}}(n)|$ be the number of codewords in $\mathcal{C}_{\boldsymbol{k}}(n)$ for UW $\boldsymbol{k}$, where $\mathcal{C}_{\boldsymbol{k}}(n)$ is the UDOOC consisting of length-n codewords (cf. Definition 1). To quantify how fast $c_{\boldsymbol{k},n}$ grows as n increases, we introduce the following asymptotic measure.

Definition 2: Given UW $\boldsymbol{k}$, the asymptotic growth rate of the corresponding UDOOC is defined as

$$g_{\boldsymbol{k}} = \lim_{n\to\infty} \frac{c_{\boldsymbol{k},n+1}}{c_{\boldsymbol{k},n}}. \tag{1}$$

It is obvious that $g_{\boldsymbol{k}} \le 2$ for all UW $\boldsymbol{k}$ as the unconstrained binary sequences have growth rate equal to 2. Notably, the limit in (1) must exist since $g_{\boldsymbol{k}}$ is actually the *spectral radius* of the digraph associated with $\boldsymbol{k}$ that will be discussed shortly.

Clearly, UDOOCs having larger growth rate mean that asymptotically they contain more codewords and achieve smaller average codeword length. To determine $g_{\boldsymbol{k}}$ for any UW $\boldsymbol{k}$, we shall investigate the enumeration of $c_{\boldsymbol{k},n}$, and for this we introduce a digraph for UDOOC.

A. Digraph for Enumerating $c_{\boldsymbol{k},n}$

For convenience of defining a digraph, we abbreviate the binary sequence $i_s i_{s+1} \dots i_t$ as i_s^t for $s \le t$, and regard i_s^t as an empty stream when $s > t$. In addition, we use the convenient shorthands $(\mathtt{A})_{i,j}$ and $(\underline{x})_j$ to denote the i,j-th entry and j-th entry of matrix $\mathtt{A}$ and vector $\underline{x}$, respectively. Denote by $G_{\boldsymbol{k}} = (V, E_{\boldsymbol{k}})$ a digraph corresponding to UW $\boldsymbol{k}$, in which $V = \mathbb{F}^{L-1}$ is the set of all binary length-$(L-1)$ tuples, and the edge set $E_{\boldsymbol{k}}$ is given by

$$E_{\boldsymbol{k}} = \left\{(\boldsymbol{i}, \boldsymbol{j}) : i_2^{L-1} = j_1^{L-2}, i_1\boldsymbol{j} \ne \boldsymbol{k}\right\}.$$

Define the 2^{L-1}-by-2^{L-1} adjacency matrix $\mathtt{A}_{\boldsymbol{k}}$ for digraph $G_{\boldsymbol{k}}$ as

$$(\mathtt{A}_{\boldsymbol{k}})_{i,j} = \begin{cases} 1, & \text{if } (\boldsymbol{i}, \boldsymbol{j}) \in E_{\boldsymbol{k}}, \\ 0, & \text{otherwise,} \end{cases}$$

where we have assumed that $\boldsymbol{i}$ (resp. $\boldsymbol{j}$) is the binary representation of index i (resp. j) for $i, j = 0, 1, \dots, 2^{L-1}-1$ with the leftmost bit being the most significant bit. We next define a length-2^{L-1} *initial vector* $\underline{x}_{\boldsymbol{k}}$ with $(\underline{x}_{\boldsymbol{k}})_j = 1$ if index j has the binary representation k_2^L, and $(\underline{x}_{\boldsymbol{k}})_j = 0$, otherwise. This vector $\underline{x}_{\boldsymbol{k}}$ marks the starting node in digraph $G_{\boldsymbol{k}}$. With the above notations, it follows that the j-th entry of $\underline{x}_{\boldsymbol{k}}^\top (\mathtt{A}_{\boldsymbol{k}})^n$ gives the number of length-n walks on digraph $G_{\boldsymbol{k}}$ that end at node $\boldsymbol{j}$ (i.e., the binary representation of j), where superscript "$\top$" stands for transpose operation.

Recall that every codeword $\boldsymbol{c}$ must be followed by UW $\boldsymbol{k}$ to separate consecutive codewords in a compressed stream. With $\boldsymbol{k} = k_1 k_2 \dots k_L$, it implies that $\boldsymbol{c}$ is a codeword if, and only if, the sequence $\boldsymbol{c}k_1^{L-1}$ is a valid walk on digraph $G_{\boldsymbol{k}}$ (which ensures that $\boldsymbol{k}$ is not an internal subword of $\boldsymbol{kck}$). Thus, $c_{\boldsymbol{k},n}$ is the number of distinct length-$(n+L-1)$ walks from node k_2^L to node k_1^{L-1} on digraph $G_{\boldsymbol{k}}$.

From the above discussion, let $\underline{y}_{\boldsymbol{k}}$ be the *ending vector* given by $(\underline{y}_{\boldsymbol{k}})_j = 1$ if index j has the binary representation k_1^{L-1}, and $(\underline{y}_{\boldsymbol{k}})_j = 0$, otherwise. We conclude

$$c_{\boldsymbol{k},n} = \underline{x}_{\boldsymbol{k}}^\top (\mathtt{A}_{\boldsymbol{k}})^{n+L-1} \underline{y}_{\boldsymbol{k}}. \tag{2}$$

Using the standard technique, we can enumerate $c_{\boldsymbol{k},n}$ using an indeterminate z as follows:

$$\sum_{n=0}^{\infty} c_{\boldsymbol{k},n} z^n = \frac{\underline{x}_{\boldsymbol{k}}^\top \operatorname{adj}(\mathtt{I} - \mathtt{A}_{\boldsymbol{k}} z)\, \mathtt{A}_{\boldsymbol{k}}^{L-1} \underline{y}_{\boldsymbol{k}}}{\det(\mathtt{I} - \mathtt{A}_{\boldsymbol{k}} z)}. \tag{3}$$

where $\mathtt{I}$ is the 2^{L-1}-by-2^{L-1} identity matrix. As a result, the enumeration of $c_{\boldsymbol{k},n}$ depends largely upon polynomial $\det(\mathtt{I} - \mathtt{A}_{\boldsymbol{k}} z)$. In particular, $\det(\mathtt{I} - \mathtt{A}_{\boldsymbol{k}} z)$ gives a linear recursion of $c_{\boldsymbol{k},n}$. Further, it can be shown that the asymptotic growth rate $g_{\boldsymbol{k}}$ equals exactly the largest eigenvalue of $\mathtt{A}_{\boldsymbol{k}}$.

B. Determining $g_{\boldsymbol{k}}$ using a Super-Code

So far, we have seen that given any UW $\boldsymbol{k}$, we can determine the asymptotic growth rate $g_{\boldsymbol{k}}$ by computing the largest eigenvalue of $\mathtt{A}_{\boldsymbol{k}}$. However, it is often difficult to find a closed-form expression for $g_{\boldsymbol{k}}$ due to the special structure of $\mathtt{A}_{\boldsymbol{k}}$. On the other hand, by noting that $g_{\boldsymbol{k}}$ is completely determined by digraph $G_{\boldsymbol{k}}$, it might be possible to determine $g_{\boldsymbol{k}}$ using a different set of words generated by $G_{\boldsymbol{k}}$ that has the same growth rate as $c_{\boldsymbol{k},n}$ and that can be more easily analyzed in combinatorics. To this end, we consider the following set for a given UW $\boldsymbol{k}$:

$$\mathcal{S}_{\boldsymbol{k}}(n) = \{\boldsymbol{b} \in \mathbb{F}^n \ : \ \boldsymbol{k} \text{ is not an internal subword of } \boldsymbol{b}\}.$$

In words, $\mathcal{S}_{\boldsymbol{k}}(n)$ is the set of distinct length-n walks on $G_{\boldsymbol{k}}$. Let $s_{\boldsymbol{k},n} = |\mathcal{S}_{\boldsymbol{k}}(n)|$ be the number of elements in $\mathcal{S}_{\boldsymbol{k}}(n)$. By an argument similar to (3), it can be shown that

$$\sum_{n=0}^{\infty} s_{\boldsymbol{k},n} z^n = \sum_{n=0}^{L-1} 2^n z^n + z^L \frac{\underline{\mathbf{1}}^\top \mathrm{adj}\,(\mathtt{I} - \mathtt{A}_{\boldsymbol{k}} z)\,\underline{\mathbf{1}}}{\det(\mathtt{I} - \mathtt{A}_{\boldsymbol{k}} z)}, \qquad (4)$$

where $\underline{\mathbf{1}}$ is the all-one vector of appropriate length. It is seen that the enumeration of $s_{\boldsymbol{k},n}$ also depends upon the same polynomial $\det(\mathtt{I} - \mathtt{A}_{\boldsymbol{k}} z)$ as $c_{\boldsymbol{k},n}$ does. Furthermore, $s_{\boldsymbol{k},n}$ has the same asymptotic growth rate as $c_{\boldsymbol{k},n}$.

Theorem 1: For any given UW $\boldsymbol{k}$, sequences $\{c_{\boldsymbol{k},n}\}$ and $\{s_{\boldsymbol{k},n}\}$ have the same asymptotic growth rate $g_{\boldsymbol{k}}$.

Proof: Clearly, $\mathcal{S}_{\boldsymbol{k}}(n)$ is a super-code for $\mathcal{C}_{\boldsymbol{k}}(n)$; hence $s_{\boldsymbol{k},n} \geq c_{\boldsymbol{k},n}$. On the other hand, we can show $c_{\boldsymbol{k},n+2} \geq s_{\boldsymbol{k},n}$ for $n \geq L$. The details are omitted due to lack of space. ■

Determining the enumeration of $s_{\boldsymbol{k},n}$ turns out to be much easier in comparison with $c_{\boldsymbol{k},n}$ due to lesser constraints on sequences in $\mathcal{S}_{\boldsymbol{k}}(n)$. One approach is to employ the *Goulden-Jackson clustering method* [5], [6]. To this end, we first define the *auto-overlap function* (AOF) of UW $\boldsymbol{k}$ as follows.

Definition 3: Given UW $\boldsymbol{k}$ of length L, the AOF of $\boldsymbol{k}$ is defined as

$$r_{\boldsymbol{k}}(i) = \begin{cases} 1 & \text{if } k_{i+1}^L = k_1^{L-i} \text{ for } 1 \leq i \leq L-1 \\ 0 & \text{otherwise.} \end{cases}$$

With the AOF $r_{\boldsymbol{k}}(i)$, the enumeration of $s_{\boldsymbol{k},n}$ is given completely by the following theorem.

Theorem 2 ([5]): Let $r_{\boldsymbol{k}}(i)$ be the AOF for the length-L UW $\boldsymbol{k}$; then

$$\sum_{n \geq 0} s_{\boldsymbol{k},n} z^n = \frac{1 + \sum_{i=1}^{L-1} r_{\boldsymbol{k}}(i) z^i}{(1-2z)\left(1 + \sum_{i=1}^{L-1} r_{\boldsymbol{k}}(i) z^i\right) + z^L}. \qquad (5)$$

Theorem 2 provides a compact description of the determinant $\det(\mathtt{I} - \mathtt{A}_{\boldsymbol{k}} z)$ for the adjacency matrix $\mathtt{A}_{\boldsymbol{k}}$ using AOF $r_{\boldsymbol{k}}(i)$. Furthermore, it can be used to derive an explicit linear recursion for $c_{\boldsymbol{k},n}$.

Theorem 3: Let $\mathtt{A}_{\boldsymbol{k}}$ be the adjacency matrix defined as before. Then

$$\det(\mathtt{I} - \mathtt{A}_{\boldsymbol{k}} z) = h_{\boldsymbol{k}}(z) = (1-2z)\left(1 + \sum_{i=1}^{L-1} r_{\boldsymbol{k}}(i) z^i\right) + z^L \qquad (6)$$

Moreover, for $n \geq L$, we have

$$c_{\boldsymbol{k},n} = \left[\sum_{i=1}^{L-1} r_{\boldsymbol{k}}(i)\,(2c_{\boldsymbol{k},n-i-1} - c_{\boldsymbol{k},n-i})\right] + 2c_{\boldsymbol{k},n-1} - c_{\boldsymbol{k},n-L}. \qquad (7)$$

Proof: The right-hand-side (RHS) of (5) is an irreducible rational function with denominator $h_{\boldsymbol{k}}(z)$; hence $h_{\boldsymbol{k}}(z)$ divides $\det(\mathtt{I} - \mathtt{A}_{\boldsymbol{k}} z)$ in ring $\mathbb{Z}[z]$. Further, we can show the degree of polynomial $\det(\mathtt{I} - \mathtt{A}_{\boldsymbol{k}} z)$ is L. Thus, $\det(\mathtt{I} - \mathtt{A}_{\boldsymbol{k}} z) = h_{\boldsymbol{k}}(z)$. The proof for (7) follows from an application of Cayley-Hamilton theorem. The details are omitted due to lack of space. ■

We have seen that $c_{\boldsymbol{k},n}$ and $s_{\boldsymbol{k},n}$ have enumerations both depending upon $\det(\mathtt{I} - \mathtt{A}_{\boldsymbol{k}} z)$ and have the same asymptotic growth rate. The enumeration of the latter is determined completely via Theorem 2. Below we will consider the asymptotic growth rates of two specific UWs, $\boldsymbol{a} = 0\ldots00$ and $\boldsymbol{b} = 0\ldots01$. It will be seen that UW $\boldsymbol{a}$ gives the largest growth rate among all UWs of length L, while interestingly UW $\boldsymbol{b}$ gives the smallest growth rate.

Theorem 4: Let $\boldsymbol{a} = 0\ldots0$ and $\boldsymbol{b} = 0\ldots01$ be two UWs of length L. Then for any UW $\boldsymbol{k}$ of length L,

$$g_{\boldsymbol{b}} \ \leq \ g_{\boldsymbol{k}} \ \leq \ g_{\boldsymbol{a}}. \qquad (8)$$

Furthermore, for sufficiently large L,

$$g_{\boldsymbol{a}} \approx 2 - 2^{-L} \quad \text{and} \quad g_{\boldsymbol{b}} \approx 2 - 2^{1-L}.$$

Theorem 4 indicates that compressing a source with large number of alphabets using UDOOC based on UW $\boldsymbol{a} = 0\ldots0$ yields the best performance. For source with small number of alphabets, however, it will be seen in Section IV that the UW $\boldsymbol{b} = 0\ldots01$ is actually a better choice. This is because the source is too small in size for asymptotics. Table I shows the values of $g_{\boldsymbol{a}}$ and $g_{\boldsymbol{b}}$ for various L. It can be verified that the approximations given in Theorem 4 are fairly accurate.

TABLE I
THE ASYMPTOTIC GROWTH RATES FOR $\boldsymbol{a}$ AND $\boldsymbol{b}$ WITH VARIOUS L

L	2	3	4	5	6	7	8
$g_{\boldsymbol{a}}$	1.618	1.839	1.928	1.966	1.984	1.992	1.996
$g_{\boldsymbol{b}}$	1	1.618	1.839	1.928	1.966	1.984	1.922

III. Encoding and Decoding Algorithms

A good source code with minimum average codeword-length should assign codewords of shorter lengths to messages with higher probabilities and reserve longer codewords only for the unlikely messages. Following this principle, let $\mathcal{U} = \{u_1, u_2, \cdots\}$ be the set of message alphabets, and let p_i be the probability that alphabet u_i occurs. Without loss of generality, we assume $p_1 \geq p_2 \geq \cdots$. Let $\mathcal{C}$ be a good lossless source code with encoding function $\phi : \mathcal{U} \to \mathcal{C}$. The above principle suggests $\ell(\phi(u_i)) \leq \ell(\phi(u_j))$ whenever $i \leq j$, where $\ell(\boldsymbol{c})$ denote the length of codeword $\boldsymbol{c}$.

To apply the above principle to UDOOC codes, given the UW $\boldsymbol{k}$, let $\mathcal{C}_{\boldsymbol{k}}$ be the corresponding UDOOC. Then,

the corresponding encoding function $\phi_{\boldsymbol{k}} : \mathcal{U} \to \mathcal{C}_{\boldsymbol{k}}$ can be designed as follows. Define

$$F_{\boldsymbol{k},n} = \begin{cases} \sum_{t=0}^{n} c_{\boldsymbol{k},t}, & \text{if } n \geq 0, \\ 0, & \text{otherwise;} \end{cases} \tag{9}$$

then $\phi_{\boldsymbol{k}}$ should be a bijection between $\{u_i : 1 \leq i \leq F_{\boldsymbol{k},n}\}$ and $\bigcup_{i=0}^{n} \mathcal{C}_{\boldsymbol{k}}(i)$ for all n, provided that the sequence $p_i = \Pr\{u_i\}$, $i = 1, 2, \cdots$, is non-increasing. Any such assignment results in the same smallest average codeword-length

$$L_{\boldsymbol{k}} = \ell(\boldsymbol{k}) + \sum_{u \in \mathcal{U}} \Pr\{u\} \ell(\phi_{\boldsymbol{k}}(u)), \tag{10}$$

where the first term $\ell(\boldsymbol{k})$ is to account for the insertion of $\boldsymbol{k}$ to separate adjacent codewords.

To provide an explicit design of $\phi_{\boldsymbol{k}}$, we need to give a bijection between the set of length-n codewords $\mathcal{C}_{\boldsymbol{k}}(n)$ and the set of message alphabets $\{u_m : F_{\boldsymbol{k},n-1} < m \leq F_{\boldsymbol{k},n}\}$. For $n \geq 1$ the proposed encoding algorithm is as follows. Given any prefix word $\boldsymbol{d}$ with $\ell(\boldsymbol{d}) \leq n$, we define

$$\mathcal{C}_{\boldsymbol{k}}(\boldsymbol{d}, n) = \{\boldsymbol{c} \in \mathcal{C}_{\boldsymbol{k}}(n) : \ \boldsymbol{d} \text{ is a prefix of } \boldsymbol{c}\}.$$

Now, given message u_m with $F_{\boldsymbol{k},n-1} < m \leq F_{\boldsymbol{k},n}$, to determine the codeword $\phi_{\boldsymbol{k}}(u_m) = \boldsymbol{c} = c_1 \dots c_n$, we introduce the following branching-metric variable

$$b_0 = m - F_{\boldsymbol{k},n-1}$$

to keep track of the branching of each coded bit; then the first bit c_1 is given by

$$c_1 = \begin{cases} 0, & \text{if } b_0 \leq |\mathcal{C}_{\boldsymbol{k}}(0, n)| \\ 1, & \text{if } b_0 > |\mathcal{C}_{\boldsymbol{k}}(0, n)|. \end{cases}$$

Similarly, having determined $\boldsymbol{d} = c_1 \dots c_{i-1}$ and metric b_{i-1}, the assignment for the next bit c_i is

$$c_i = \begin{cases} 0, & \text{if } b_{i-1} \leq |\mathcal{C}_{\boldsymbol{k}}(\boldsymbol{d}0, n)|, \\ 1, & \text{if } b_{i-1} > |\mathcal{C}_{\boldsymbol{k}}(\boldsymbol{d}0, n)|. \end{cases}$$

The update for b_{i-1} is

$$b_i = \begin{cases} b_{i-1}, & \text{if } c_i = 0, \\ b_{i-1} - |\mathcal{C}_{\boldsymbol{k}}(\boldsymbol{d}0, n)|, & \text{if } c_i = 1. \end{cases}$$

It can be shown that at the end of the nth iteration, say $\phi_{\boldsymbol{k}}(u_m) = \boldsymbol{c} = c_1 \dots c_n$, we must have

$$m = F_{\boldsymbol{k},n-1} + \sum_{\substack{i:1 \leq i < n \\ c_i = 1}} \left|\mathcal{C}_{\boldsymbol{k}}(c_1^{i-1}0, n)\right| + 1 \tag{11}$$

hence, $\phi_{\boldsymbol{k}}$ is indeed a bijection between $\{u_m : F_{\boldsymbol{k},n-1} < m \leq F_{\boldsymbol{k},n}\}$ and $\mathcal{C}_{\boldsymbol{k}}(n)$. We shall emphasize that (11) also gives the corresponding decoding function $\psi_{\boldsymbol{k}} : \mathcal{C}_{\boldsymbol{k}} \to \mathcal{U}$ for $\boldsymbol{c}$. Specifically, given codeword $\boldsymbol{c} \in \mathcal{C}_{\boldsymbol{k}}$ with $\ell(\boldsymbol{c}) = n$, the decoding function is exactly $\psi_{\boldsymbol{k}}(\boldsymbol{c}) = u_m$, where m is given by (11).

In the above encoding and decoding algorithms, although $c_{\boldsymbol{k},i}$ and $F_{\boldsymbol{k},i}$ can be determined by (2) and (9), we still need to determine $|\mathcal{C}_{\boldsymbol{k}}(\boldsymbol{d}0, n)|$ for any prefix $\boldsymbol{d}$. For some specific UWs, such as $\boldsymbol{a} = 0 \dots 0$, $\boldsymbol{b} = 0 \dots 01$, and their binary complements, the values of $|\mathcal{C}_{\boldsymbol{k}}(\boldsymbol{d}0, n)|$ and $c_{\boldsymbol{k},n}$ can be easily determined from the following theorems. Notably, it is clear that the algorithms for encoding or decoding UDOOCs with UW $\boldsymbol{a} = 0 \dots 0$ or $\bar{\boldsymbol{a}} = 1 \dots 1$ are equivalent, as one can be obtained from the other by binary complementing. The same also holds for the case of $\boldsymbol{b} = 00 \dots 01$ or $\bar{\boldsymbol{b}} = 1 \cdots 10$. Thus, here we focus only on the cases of $\boldsymbol{k} = \bar{\boldsymbol{a}} = 1 \dots 1$ and $\boldsymbol{k} = \bar{\boldsymbol{b}} = 1 \cdots 10$ with $\ell(\boldsymbol{k}) = L$.

Theorem 5: Let $\boldsymbol{k} = \bar{\boldsymbol{a}} = 1 \dots 1$ with $\ell(\boldsymbol{k}) = L$. Then given any prefix $\boldsymbol{d}$ with $|\mathcal{C}_{\boldsymbol{k}}(\boldsymbol{d}0, n)| > 0$, we have $|\mathcal{C}_{\boldsymbol{k}}(\boldsymbol{d}0, n)| = c_{\boldsymbol{k},n-\ell(\boldsymbol{d})}$. Furthermore, the values of $c_{\boldsymbol{k},n}$ are given by

$$c_{\boldsymbol{k},n} = \begin{cases} 1, & n = 0, 1, 2, \\ 2^{n-2}, & n = 3, \dots, L \\ \sum_{i=1}^{L} c_{\boldsymbol{k},n-i} & n > L. \end{cases} \tag{12}$$

Theorem 6: Let $\boldsymbol{k} = \bar{\boldsymbol{b}} = 1 \cdots 10$ with $\ell(\boldsymbol{k}) = L$. Then given any prefix $\boldsymbol{d}$ with $|\mathcal{C}_{\boldsymbol{k}}(\boldsymbol{d}0, n)| > 0$, we have $|\mathcal{C}_{\boldsymbol{k}}(\boldsymbol{d}0, n)| = c_{\boldsymbol{k},n-\ell(\boldsymbol{d})-1}$. Furthermore, the values of $c_{\boldsymbol{k},n}$ are given by

$$c_{\boldsymbol{k},n} = \begin{cases} 2^n, & n = 0, 1, \dots, L-1 \\ 2c_{\boldsymbol{k},n-1} - c_{\boldsymbol{k},n-L}, & n \geq L. \end{cases} \tag{13}$$

To sum up, in Algorithms 1 and 2 we present the respective encoding and decoding functions for the UDOOCs with $\boldsymbol{k} = \bar{\boldsymbol{a}} = 1 \dots 1$ or $\boldsymbol{k} = \bar{\boldsymbol{b}} = 1 \dots 10$. We assume that the set of message alphabets $\mathcal{U} = \{u_1, u_2, \dots\}$ are ordered according to their probability masses, that is, $\Pr\{u_i\} \geq \Pr\{u_j\}$ if $i < j$.

Algorithm 1 Encoding of $\mathcal{C}_{\boldsymbol{k}}$ with $\boldsymbol{k} = \bar{\boldsymbol{a}}$ or $\boldsymbol{k} = \bar{\boldsymbol{b}}$

Input: Index m for message u_m
Output: Codeword $\phi_{\boldsymbol{k}}(u_m) = c_1 \dots c_n$

```
Repeatedly compute c_{k,0}, c_{k,1}, ..., c_{k,n} using (12) for k = ā (or (13) for k = b̄ ) to find the smallest n such that F_{k,n} ≥ m
b_0 ← m − F_{k,n−1}
for i = 1 to n do
    B ← c_{k,n−i+1} for k = ā or B ← c_{k,n−i} for k = b̄
    if b_{i−1} ≤ B then
        c_i ← 0 and b_i ← b_{i−1}
    else
        c_i ← 1 and b_i ← b_{i−1} − B
    end if
end for
```

IV. Performance of UDOOC

In Fig. 2, we plot the values of $c_{\boldsymbol{k},n}$ for various $\boldsymbol{k}$ of length $L = 3$. The values are presented in log-scale and are normalized against the case of $\boldsymbol{k} = 0 \dots 0$. It should be noted that we only show the curves for inequivalent UWs, by which we mean that $\boldsymbol{k}$ is equivalent to $\boldsymbol{k}'$ if $c_{\boldsymbol{k},n} = c_{\boldsymbol{k}',n}$ for all n. We observe that $c_{\boldsymbol{k},n}$ for $\boldsymbol{k} = 0 \dots 01$ is the largest among all when n is small, but it becomes the smallest when n is large. $c_{\boldsymbol{k},n}$ for $\boldsymbol{k} = 0 \dots 0$ is the smallest among all when n is small, but it is the largest when n is large. This shows a tradeoff among the

Algorithm 2 Decoding of $\mathcal{C}_{\boldsymbol{k}}$ with $\boldsymbol{k} = \bar{\boldsymbol{a}}$ or $\boldsymbol{k} = \bar{\boldsymbol{b}}$

Input: Codeword $\boldsymbol{c} = c_1 \dots c_n$
Output: Index m for message $u_m = \psi_{\boldsymbol{k}}(\boldsymbol{c})$

```
Compute c_{k,0}, c_{k,1}, ..., c_{k,n} using (12) for k = ā (or (13) for k = b̄)
m ← F_{k,n-1} + 1
for i = 1 to n do
    B ← c_{k,n-i+1} for k = ā or B ← c_{k,n-i} for k = b̄
    if c_i = 1 then
        m ← m + B
    end if
end for
```

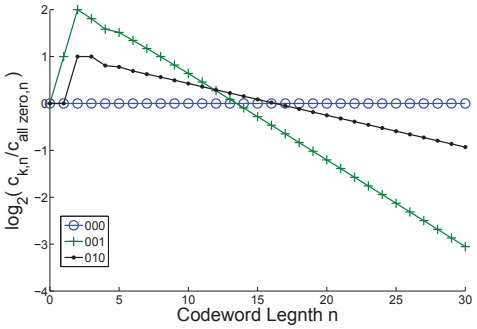

Fig. 2. Normalized number of length-n codewords for $L = 3$

choices of UWs. If the source is uniform and is extremely large in size, UDOOC using $\boldsymbol{k} = 0 \dots 0$ achieves better compression performance and has smaller average codeword length. On the other hand, if the source is nonuniform and is smaller in size, $\boldsymbol{k} = 0 \dots 01$ or its equivalence is a better choice.

We also simulate the compression performance of UDOOCs when source is uniform and independent distributed among 26 English letters [7]. The results are given in Table II, where the average codeword length has already taken into account the length of UWs. We observe that when the parser size $M = 1$, meaning the alphabet size is small, UDOOC with $\boldsymbol{k} = 01$ gives the best performance. However, as M increases, the alphabet size grows exponentially, and compression performance using $\boldsymbol{k} = 0 \dots 0$ eventually becomes the best. It should also be noted that for large M UDOOC with UW 01 performs extremely poor due to its asymptotic growth rate equal to 1.

TABLE II
AVERAGE CODEWORD LENGTH IN BITS PER SOURCE SYMBOL FOR UNIFORM INDEPENDENT 26 ENGLISH LETTERS WITH PARSER SIZE M

	UW = $00 \cdots 0$				**UW =** $00 \cdots 01$			
	M=1	M=2	M=4	M=8	M=1	M=2	M=4	M=8
L=2	6.961	6.820	6.794	6.781	5.846	12.76	159	53855
L=4	8.576	6.830	5.901	5.434	7.000	5.898	5.619	5.474
L=8	12.576	8.735	6.718	5.719	11.000	7.751	6.222	5.477

We next consider two different sources. The first is an independent English source with usual frequency [7]. The second source is the book "Alice's Adventures in Wonderland" [8], which represents a dependent source with memory. Results of the compression performance using UDOOCs are presented in Table III and are compared with those using the conventional Huffman and Lempel-Ziv (LZ) codes [2]. It is seen that with parser size $M = 3$, all UDOOCs with $\boldsymbol{k} = 0 \dots 0$ perform better than LZ code.

TABLE III
AVERAGE CODEWORD LENGTH IN BITS PER SOURCE SYMBOL FOR THE COMPRESSION OF TWO DIFFERENT SOURCES

(a)

Type	**Entropy**			**LZ**	**Huffman**		
	M=1	M=2	M=3		M=1	M=2	M=3
Usual Independent English Letters	4.246	4.246	4.246	6.626	4.274	4.261	4.253
	M=1	M=2	M=3		M=1	M=2	M=3
Alice's Adventures in Wonderland	3.914	3.570	3.215	6.027	3.940	3.585	3.226

(b)

Type		**UW =** $00 \cdots 0$			**UW =** $00 \cdots 01$		
		M=1	M=2	M=3	M=1	M=2	M=3
Usual Independent English Letters	L=2	5.591	5.550	5.637	4.557	7.771	20.907
	L=4	7.411	5.872	5.351	6.185	4.970	4.795
	L=6	9.411	6.818	5.924	8.185	5.882	5.274
		M=1	M=2	M=3	M=1	M=2	M=3
Alice's Adventures in Wonderland	L=2	4.887	4.340	3.958	4.068	4.975	7.573
	L=4	6.757	4.920	4.089	5.774	4.133	3.531
	L=6	8.757	5.890	4.709	7.774	5.089	4.115

V. CONCLUSION

In this paper, we proposed a class of UDOOC that allows for a fast identification of all codewords in a compressed stream, and therefore parallel decoding is possible. Many combinatorial properties of UDOOC, such as the number of length-n codes, enumerations, and asymptotic growth rate, are presented and are helpful in choosing appropriate UWs to yield better compression performance. Encoding and decoding algorithms for UDOOC associated with UW $0 \dots 0$ and $0 \dots 01$ are given and can be implemented without storing the codebook. We simulated the compression performance using UDOOC and demonstrated that in many cases the UDOOC performs better than the LZ code.

REFERENCES

[1] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, July, October 1948.
[2] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, New York, NY: John Wiley & Sons, 1991.
[3] W. Szpankowski, "One-to-one codes and its anti-redundancy," in Proceedings *IEEE ISIT*, September 4-9, 2005.
[4] D. B. West , *Introduction to Graph Theory (2E)*, Prentice Hall, 2001.
[5] I. Goulden and D. M. Jackson, "An inversion theorem for cluster decompositions of sequences with distinguished subsequences," *J. London Math. Soc.*, vol. 20, no. 2, pp. 567-576, 1979.
[6] J. Noonan and D. Zeilberger, "The Goulden-Jackson cluster method: extensions, applications, and implementations," *J. Difference Eq. Appl.*, vol. 5, pp. 355-377, 1999.
[7] http://oxforddictionaries.com/words/what-is-the-frequency-of-the-letters-of-the-alphabet-in-english
[8] C. L. Dodgson (L. Carroll), "Alice's Adventures in Wonderland," 1865.