Zero-Error Multichannel Source Coding

Hongyi Yao and Raymond W. Yeung, *Fellow, IEEE*

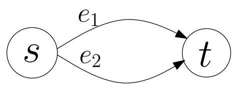

Fig. 1. A two-channel communication system.

***Abstract*—We introduce the problem of zero-error multichannel source coding, in which a transmitter needs to transmit an information source to a receiver with zero error through possibly more than one error-free channel. We present the basic results related to this problem.**

I. Introduction

The traditional source coding problem concerns the transmission of an information source through a single channel from the transmitter to the receiver. When multiple channels are involved, the problem may become more complicated. To illustrate the idea, we consider the network in Fig. 1 discussed in [5]. In this network, the two edges e_1 and e_2 represents error-free channels 1 and 2 from node s to node t, respectively. Consider transmitting an information source X from s to t through channels 1 and 2. Let the alphabet of X be $\mathcal{X} = \{1, 2, 3, 4, 5, 6\}$, and suppose the source symbols 1, 2, 3, 4, 5 and 6 are encoded by encoders dedicated to channels 1 and 2 into

$$0, 1, 00, 01, 10, 11, \tag{1}$$

and

$$0, 0, 1, 1, 1, 1, \tag{2}$$

respectively.

Let U_1 and U_2 be the random codewords sent on channels 1 and 2, respectively. Equation (1) gives the collection of values (not necessarily distinct) that can be taken by U_1, which do not form a prefix-free set. Likewise, equation (2) gives the collection of values that can be taken by U_2, which again do not form a prefix-free set. However, conditioning on the value of U_2 which has length 1 with probability 1, the values that can be taken by U_1 form a prefix-free set. For example, when U_2 takes the value 1, the values that can be taken by U_1 are 00, 01, 10, 11, which form a prefix-free set. So at node t, the information source X can be recovered with zero error.

In this paper, we formulate the problem of zero-error multichannel source coding. The results presented here are generalizations of the classical results in zero-error source coding (single-channel), of which a comprehensive treatment can be found in [6, Ch. 4].

II. Multichannel Source Codes

Let a symbol transmitted on a channel be taken from the q-ary alphabet $Q = \{1, 2, \ldots, q\}$. Let channels 1 to k be the channels connecting node s and node t. Denote the alphabet of the information source X by $\mathcal{X}$, and the q-ary entropy of X by $H_q(X)$.

Definition 1 (Multichannel source code): A q-ary, k-channel source code $\mathcal{C}$ for a source random variable X is a mapping from $\mathcal{X}$ to $(Q^*)^k$, where Q^* is the set of all finite length sequences of symbols taken from the q-ary code alphabet Q and $(Q^*)^k = \{(Y_1, Y_2, \ldots, Y_k) : Y_i \in Q^*\}$. For $x \in \mathcal{X}$, $\mathcal{C}(x)$ is called the codeword for x, and the image of $\mathcal{C}$ is called the codebook.

Consider an information source sequence $\{X_i : i \geq 1\}$, where X_i are discrete random variables taking values in a common alphabet. We apply a k-channel source code $\mathcal{C}$ to each X_i, where the codewords are concatenated channelwise. Once the codewords transmitted on the channels are concatenated, the boundaries of the codewords are no longer explicit. In other words, when the code $\mathcal{C}$ is applied to a source sequence, k sequences of code symbols are produced, and the codewords may no longer be distinguishable. We are particularly interested in uniquely decodable codes which are defined as follows.

Definition 2 (Uniquely decodable code): A k-channel code $\mathcal{C}$ is uniquely decodable if for any finite source sequence, the k sequences of code symbols corresponding to this source sequence is different from the k sequences of code symbols corresponding to any other (finite) source sequence.

Next, we define a class of uniquely decodable codes which has the desirable property that a codeword can be recognized instantaneously once its transmission on all the k channels is completed. In other words, a codeword can be decoded without referring to the symbols of any future codewords.

Definition 3 (Self-punctuating code): A k-channel code

$$\mathcal{C} = \{(Y_1^{(1)}, Y_2^{(1)}, \ldots, Y_k^{(1)}), (Y_1^{(2)}, Y_2^{(2)}, \ldots, Y_k^{(2)}), \ldots\}$$

is *self-punctuating* if for any k sequences of symbols $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k$ generated by encoding a finite source sequence with the code $\mathcal{C}$, there exists exactly one sequence tuple $(\mathcal{S}'_1, \mathcal{S}'_2, \ldots, \mathcal{S}'_k)$ such that $(\mathcal{S}'_1, \mathcal{S}'_2, \ldots, \mathcal{S}'_k)$ is a codeword in $\mathcal{C}$ and $\mathcal{S}'_i$ is a prefix of $\mathcal{S}_i$ for all i.

It's well known that for the single channel case, a code $\mathcal{C}$ is self-punctuating if and only if $\mathcal{C}$ is a prefix-free code. However, it is more complicated for the multichannel case. We first introduce a stronger notion of a self-punctuating code.

Definition 4 (Strongly self-punctuating code): A k-channel code

$$\mathcal{C} = \{(Y_1^{(1)}, Y_2^{(1)}, \ldots, Y_k^{(1)}), (Y_1^{(2)}, Y_2^{(2)}, \ldots, Y_k^{(2)}), \ldots, \}$$

is strongly self-punctuating, if for any k finite sequences of symbols $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k$ (not necessarily generated by encoding a finite source sequence), there exists no more than one sequence tuple $(\mathcal{S}'_1, \mathcal{S}'_2, \ldots, \mathcal{S}'_k)$ such that $(\mathcal{S}'_1, \mathcal{S}'_2, \ldots, \mathcal{S}'_k)$ is a codeword and $\mathcal{S}'_i$ is a prefix of $\mathcal{S}_i$ for all i.

There exist codes which are self-punctuating but not strongly self-punctuating. For example, let $\mathcal{C} = \{(011, 10), (01, 100)\}$. Upon considering all the four combinations when two codewords are concatenated, we conclude that $\mathcal{C}$ is self-punctuating. However, for the two symbol sequences $\mathcal{S}_1 = 011$ and $\mathcal{S}_2 = 100$, either $(011, 10)$ or $(01, 100)$ can be the leading codeword in $(\mathcal{S}_1, \mathcal{S}_2)$.

Definition 5 (Multichannel prefix-free code): Two k-channel codewords $(Y_1^{(1)}, Y_2^{(1)}, \ldots, Y_k^{(1)})$ and $(Y_1^{(2)}, Y_2^{(2)}, \ldots, Y_k^{(2)})$ are *prefix-free* if there exists j such that $Y_j^{(1)}$ is not a prefix of $Y_j^{(2)}$

Hongyi Yao is with the Institute of Theoretical Computer Science of Tsinghua University, Beijing, P.R China, 100084. The work of Hongyi Yao was partially supported by the National Natural Science Foundation of China Grant 60553001 and the National Basic Research Program of China Grants 2007CB807900 and 2007CB807901. Email: yaohongyi03@gmail.com

Raymond Yeung is with the Department of Information Engineering, The Chinese University of Hong Kong. The work of Raymond Yeung was partially supported by a grant from the Research Grant Committee of the Hong Kong Special Administrative Region, China (RGC Ref. No. CUHK 2/06C). Email: whyeung@ie.cuhk.edu.hk

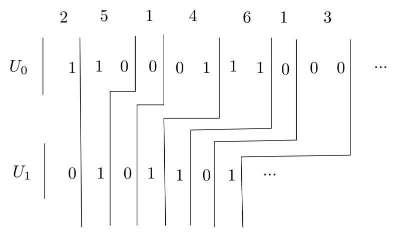

Fig. 2. The codeword sequence for the source sequence $2514613\ldots$

and $Y_j^{(2)}$ is not a prefix of $Y_j^{(1)}$. A k-channel code $\mathcal{C}$ is a *prefix-free code* if any two codewords in $\mathcal{C}$ are prefix-free.

For simplicity, we will refer to a prefix-free code as a *prefix code*. Different from the single source case, there exist codes which are self-punctuating but not prefix. For example, the code $\mathcal{C} = \{(011, 10), (01, 100)\}$ which we have seen is self-punctuating but not prefix. However, a strongly self-punctuating code is a multichannel prefix code, and vice versa, as we prove in the next theorem.

Theorem 1: A k-channel code $\mathcal{C}$ is a strongly self-punctuating code if and only if it is a prefix code.

Proof: We first prove the "if" part. If $\mathcal{C}$ is not a prefix code, there exist two codewords $(Y_1, Y_2, ..., Y_k)$ and $(Z_1, Z_2, ..., Z_k)$ in $\mathcal{C}$ such that Y_i is a prefix of Z_i or Z_i is a prefix of Y_i for all i. For all i, let $\mathcal{S}_i$ be the longer of Y_i and Z_i. Then by Definition 4, $\mathcal{C}$ is not strongly self-punctuating.

We now prove the "only if" part. If $\mathcal{C}$ is not strongly self-punctuating, then there exist two codewords $(Y_1, Y_2, ..., Y_k)$ and $(Z_1, Z_2, ..., Z_k)$ in $\mathcal{C}$ and a sequence tuple $(\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_k)$ such that both Y_i and Z_i are prefices of $\mathcal{S}_i$ for all i. Then for all i, either Y_i is a prefix of Z_i or Z_i is a prefix of Y_i, i.e., $(Y_1, Y_2, ..., Y_k)$ and $(Z_1, Z_2, ..., Z_k)$ are not prefix free. Hence, $\mathcal{C}$ is not a prefix code. The proof is completed. ■

In the example in Section I, the source symbols 1, 2, 3, 4, 5, and 6 are encoded into the codewords $(0, 0)$, $(1, 0)$, $(00, 1)$, $(01, 1)$, $(10, 1)$, and $(11, 1)$, respectively. Obviously, these codewords form a prefix code. For the source sequence 2514613..., Fig. 2 illustrates the boundaries of the codewords after they have been concatenated.

In the following, we present a decoding scheme called the Self-Punctuating Decoding Process (SPDP) for any self-punctuating code. The input to SPDP are k sequences of symbols generated by encoding a finite source sequence.

(i) For each channel, reset the pointer to the beginning of the next codeword to be decoded. If there is no more codeword to be decoded, STOP.

(ii) Set $T_1, T_2, ..., T_k$ to be the empty sequence and set $\mathcal{N}$ to be the set containing all the codewords in $\mathcal{C}$.

(iii) For i from 1 to k, read a symbol, say x_i, from channel i, move the pointer one step forward, and append x_i to T_i. If no more symbols can be read on a channel, skip that channel.

(iv) We say a codeword $(Y_1, Y_2, ..., Y_k)$ is incompatible with $(T_1, T_2, ..., T_k)$ if and only if there exists i such that Y_i is not a prefix of T_i and T_i is not a prefix of Y_i. Delete all incompatible codewords in $\mathcal{N}$.

(v) If only one codeword remains in $\mathcal{N}$, output that codeword and go back to (i); otherwise, go back to (iii).

Next, we prove the correctness of SPDP.

Theorem 2: Let $(S_1, S_2, ..., S_k)$ be a sequence tuple obtained by encoding a source sequence with a self-punctuating code $\mathcal{C}$. Then SPDP always decodes correctly.

Proof: Let Y be any codeword in $\mathcal{C}$ and suppose Y is the leading codeword in $(S_1, S_2, ..., S_k)$. Because of this, in the execution of (iv), Y cannot be deleted from $\mathcal{N}$. If the size of $\mathcal{N}$ is eventually reduced to 1, then the codeword Y will be decoded correctly. Therefore, if suffices to show that the size of $\mathcal{N}$ is always reduced to 1 after at most s iterations of (iv), where s is the length of the longest sequence among Y_i, $1 \leq i \leq k$.

If the size of $\mathcal{N}$ is reduced to 1 after $t < s$ iterations of (iv), the theorem is proved. If not, after s iterations of (iv), there exists another codeword $Y' = (Y_1', Y_2', \ldots, Y_k') \in \mathcal{N}$. Since $\mathcal{C}$ is self-punctuating, there exists i such that Y_i is not a prefix of Y_i' and Y_i' is not a prefix of Y_i. However, since Y' has not been deleted from $\mathcal{N}$ after $|Y_i| \leq s$ iterations of (iv), where $|Y_i|$ denotes the length of Y_i, Y_i is a prefix of Y_i' or Y_i' is a prefix of Y_i, which is a contradiction. Hence, the size of $\mathcal{N}$ is always reduced to 1 after at most s iterations of (iv). The theorem is proved. ■

III. Kraft Inequality and Entropy Bound

Similar to single source coding, the Kraft inequality gives a necessary condition for a code $\mathcal{C}$ to be uniquely decodable.

Theorem 3: Let $\mathcal{C}$ be a q-ary k-channel source code with m codewords. For the ith codeword $(Y_1^{(i)}, Y_2^{(i)}, \ldots, Y_k^{(i)}) \in \mathcal{C}$, denotes its length tuple by $[l_1^i, l_2^i, \ldots, l_k^i]$, where l_j^i is the length of $Y_j^{(i)}$. If $\mathcal{C}$ is uniquely decodable, then its set of length tuples satisfies

$$\sum_{i=1}^{m} q^{-s_i} \leq 1, \tag{3}$$

where

$$s_i = l_1^i + l_2^i + ... + l_k^i \tag{4}$$

is the total length of the ith codeword over all the k channels.

Proof: Let N be an arbitrary positive integer. By collecting all the terms in $(\sum_{i=1}^m q^{-s_i})^N$, we write

$$\left(\sum_{i=1}^{m} q^{-s_i}\right)^N = \sum_{j=1}^{Ns_m} A_j q^{-j},$$

where s_m is the maximum of $\{s_i\}$ and A_j is the coefficient of q^{-j} in $(\sum_{i=1}^m q^{-s_i})^N$. Now observe that A_j gives the total number of sequences of N codewords with a total length of j code symbols. Since the code is uniquely decodable, these code sequences must be distinct, and therefore

$$A_j \leq (j + k - 1)^{k-1} q^j,$$

because there are q^j distinct sequences of j symbols and at most $(j + k - 1)^{k-1}$ ways to divide them into k groups. Then we have

$$\begin{aligned} \left(\sum_{i=1}^{m} q^{-s_i}\right)^N &\leq \sum_{j=1}^{Ns_m} (j + k - 1)^{k-1} \\ &\leq (k + Ns_m)^k N s_m, \end{aligned}$$

or

$$\sum_{i=1}^{m} q^{-s_i} \leq ((k + Ns_m)^k N s_m)^{1/N}.$$

Since this inequality holds for any N, letting $N \to \infty$, we obtain (3). This can be seen by taking logarithm of the right hand side above. The theorem is proved. ■

Different from the single-channel case, the Kraft inequality is not a sufficient condition for the existence of a prefix code, and not even for the existence of a self-punctuating code. For example, $\Lambda = \{[1, 1], [1, 1], [1, 1], [1, 2], [2, 1]\}$ satisfies the Kraft inequality but we will see that there exists no binary self-punctuating code with such a set of length tuples.

We prove the nonexistence of a binary self-punctuating code with the set of length tuples given by Λ by attempting to construct such a code. There are four patterns for the first bits of the two channels, namely $(0,0)$, $(0,1)$, $(1,0)$, and $(1,1)$. The three codewords with length tuples $[1,1]$, $[1,1]$, $[1,1]$ use three of them, otherwise the code cannot be self-punctuating, and we let (a,b) with $a,b \in \{0,1\}$ be the one which has not been used. Then the codeword with length tuple $[1,2]$ must have the form (a,bx) with $x \in \{0,1\}$, otherwise, the code cannot be self-punctuating. For the same reason, the codeword with length tuple $[2,1]$ must have the form (ay,b) with $y \in \{0,1\}$. Since a valid pair of code sequences may begin with (ay,bx), we conclude that the code cannot be self-punctuating because either (a,bx) or (ay,b) can be the first codeword.

As discussed in Section I, in a k-channel self-punctuating code, the set of codewords for an individual channel may not form a prefix set. As such, the entropy bound may not apply to the expected coding length for an individual channel. For instance, consider the example in Section I (cf. Fig. 1) and let the probability mass function of X be

$$p(1) = p(2) = \frac{a}{2}, \tag{5}$$

$$p(3) = p(4) = p(5) = p(6) = \frac{1-a}{4}, \tag{6}$$

where $0 < a < 1$. Then the expected length of U_1 is given by

$$L_1 = 2\left(1 \cdot \frac{a}{2}\right) + 4\left(2 \cdot \frac{1-a}{4}\right) = 2 - a,$$

while with $q = 2$, the entropy of U_1 is given by

$$\begin{aligned} H_2(U_1) &= 2\left(-\frac{a}{2}\log_2\frac{a}{2}\right) + 4\left(-\frac{1-a}{4}\log_2\frac{1-a}{4}\right) \\ &= h_2(a) + 2 - a, \end{aligned}$$

where $h_2(a) = -a\log_2(a) - (1-a)\log_2(1-a)$. Since $h_2(a) > 0$, we have $H_2(U_1) > L_1$.

Nevertheless, since by Theorem 3, for any uniquely decodable code, $\{s_i\}$, the set of total codeword lengths, satisfies the Kraft inequality, it can be shown by standard techniques that the entropy bound applies to the expected total codeword length. This result is stated in the next theorem. We refer the reader to [6, Theorem 4.6], for example, for the details of the proof.

Theorem 4 (Entropy bound): Let $\mathcal{C}$ be a q-ary, k-channel uniquely decodable code for a source random variable X with probability $\{p_1, p_2, \ldots, p_m\}$ and entropy $H_q(X)$. Then

$$L \geq H_q(X), \tag{7}$$

where $L = \sum_{i=1}^{m} p_i s_i$. This lower bound is tight if and only if $s_i = -\log_q p_i$ for all i.

IV. Decoding Tree

For the single-channel case, a prefix code can be uniquely represented by a tree with the leaves being the codewords. This representation not only is important for understanding the structure of a prefix code but also is useful for decoding such a code. For this reason, the tree representing a prefix code is also called the *decoding tree* of the code. In fact, the existence of a decoding tree for a prefix code implies that the code can be decoded sequentially, i.e., a codeword can be decoded without referring to the symbols of any future codewords.

For the multichannel case, there is no natural tree representation of a prefix code in general. This explains why the decoding algorithm SPDP in Section II may need to refer to the symbols of future codewords. In such situations, the algorithm also needs to backtrack the pointers before decoding the next codeword.

In this section, we introduce a class of multichannel source codes that can be decoded sequentially. Such a code, called a *tree-decodable code*, is specified by a decoding tree for the code. Except for the single channel case, a tree-decodable code can have more than one decoding tree.

We now describe how a q-ary decoding tree $\mathcal{T}$ specifies a q-ary k-channel source code $\mathcal{C}$. Each internal node in $\mathcal{T}$ belongs to a *class* ranging from 1 to k that corresponds to one of the k channels. For an internal node u, denote its class by i_u. The edges connecting each internal node and its children are labeled from 1 to q, and if the internal node belongs to class i, we say that these edges belong to class i. The codeword $C_l = (c_1, c_2, ..., c_k)$ represented by a leaf l, determined by the unique path $\mathcal{P}$ from the root r to l, is such that for any i, c_i is the sequence of labels of the edges along $\mathcal{P}$ that belong to class i.

Later on, we will show that the code specified by a decoding tree is a prefix code, so that it is also strongly self-punctuating (cf. Theorem 1). We now describe the decoding scheme associated with a decoding tree:

(i) Decoding starts at the root r of the decoding tree.
(ii) Check the next symbol on channel i_r and go down the edge labeled by that symbol.
(iii) Arrive at a node u;
(iv) If node u is not a leaf, check the next symbol on channel i_u, go down the edge labeled by that symbol, and repeat (iii).
(v) Otherwise, output the codeword represented by node u.

The probability that a node u in a decoding tree is reached during the decoding process is called the *reaching probability* of u, denoted by p_u. If u is an internal node, then p_u is equal to the total probability of all the leaves (codewords) descending from u. If u is a leaf, then p_u is simply the probability of the corresponding codeword. The concept of reaching probability has been used in the literature for obtaining lower and upper bounds on the expected length of different classes of tree codes [2], [3], [4].

As before, denote the expected coding length for channel i by L_i. The following theorem expresses L_i in terms of the reaching probabilities of the internal nodes of the decoding tree.

Theorem 5: For a tree-decodable code,

$$L_i = \sum_{u \in \text{class } i} p_u$$

for $1 \leq i \leq k$.

Proof: Consider a leaf l and an internal node u. Let a_u^l be 1 if l is a descendent of u, and be 0 otherwise. Let

$$s_i^l = \sum_{u \in \text{class } i} a_u^l.$$

Recall that $C_l = (c_1, c_2, \ldots, c_k)$ is the codeword represented by leaf l. It is not difficult to see that s_i^l is equal to the length of c_i. On the other hand, by denoting the set of all leaves by $\mathcal{L}$, for each internal node u, we have

$$p_u = \sum_{l \in \mathcal{L}} a_u^l \, p_l,$$

It follows that

$$\begin{aligned} L_i &= \sum_{l \in \mathcal{L}} p_l \, s_i^l \\ &= \sum_{l \in \mathcal{L}} p_l \sum_{u \in \text{class } i} a_u^l \\ &= \sum_{u \in \text{class } i} \sum_{l \in \mathcal{L}} a_u^l \, p_l \\ &= \sum_{u \in \text{class } i} p_u. \end{aligned}$$

The theorem is proved. ■

When $k = 1$, Theorem 5 becomes Lemma 4.20 in [6], where L_1 becomes the expected length of the prefix code.

The code in the example in Section I has a decoding tree as shown in Fig. 3, where the leaves are labeled from 1 to 6. With the probability mass function of the source X as given in (5) and (6), the expected coding lengths for the two channels are

$$L_1 = \frac{1-a}{2} + \frac{1-a}{2} + (1-a) + a = 2 - a.$$

and $L_2 = 1$.

We say that a code is sequentially decodable if a codeword can be decoded by reading the symbols transmitted on the channels in sequence (i.e., without skipping symbols) and without referring to the symbols of any future codewords. We prove in the following theorem the equivalence between a tree-decodable code and a sequentially decodable code. For the single channel case, this equivalence is obvious.

Theorem 6: A code $\mathcal{C}$ is tree decodable if and only if it is sequentially decodable.

Proof: The "only if" part is trivial, so we only prove the "if" part. Let $\mathcal{C}$ be any q-ary k-channel sequentially decodable code and $\mathcal{S}$ be a sequential decoding strategy for the code. Without loss of generality, assume $\mathcal{S}$ reads one symbol on a particular channel at a time.

In the following, we construct a decoding tree corresponding to $\mathcal{S}$. Initially, let the decoding tree contain only the root r and let $\mathcal{N}$ be the empty set.

(i) Suppose the first symbol read by $\mathcal{S}$ is on channel i_r. In the decoding tree, let r belong to class i_r, add q children to r, and label the edges connecting r and its children from 1 to q. The lth child of r corresponds to the state of $\mathcal{S}$ after reading the first symbol if the value of the first symbol is l. Add all the children of r to $\mathcal{N}$.

(ii) Take any node, say u, in $\mathcal{N}$.

(ii.a) If $\mathcal{S}$ outputs a codeword at the state corresponding to node u, make node u a leaf.

(ii.b) Otherwise, suppose the next symbol read by $\mathcal{S}$ is on channel i_u. Let u belong to class i_u, add q children to u, and label the edges connecting u and its children from 1 to q. The lth child of u corresponds to the state of $\mathcal{S}$ after reading the next symbol at the state corresponding to node u if the value of that symbol is l. Add all the children of u to $\mathcal{N}$ and remove u from $\mathcal{N}$.

(iii) Repeat (ii) until $\mathcal{N}$ is empty.

We now explain the one-to-one correspondence between the decoding tree constructed above and the sequential decoding strategy $\mathcal{S}$. In the decoding tree, each node corresponds to a state of $\mathcal{S}$. Suppose $\mathcal{S}$ is at an internal node u belonging to class i_u. Then upon reading the next symbol on channel i_u that takes the value l, $\mathcal{S}$ changes state to the lth child of u.

Note that when the decoder goes down the decoding tree during the decoding process, the path from the root r to the current state uniquely specifies the sequences of symbols read on the k channels so far. Since $\mathcal{S}$ decodes a codeword correctly without referring to the symbols of any future codewords, once a codeword is completely read, $\mathcal{S}$ must output that codeword immediately. Therefore, the above construction of the decoding tree always stops in finite time. The theorem is proved. ■

Theorem 7: A tree-decodable code is a prefix code, but the converse is not true.

Proof: For any two leaves of a k-channel decoding tree $\mathcal{T}$, where the paths from the root to them are $\mathcal{P}_1$ and $\mathcal{P}_2$, there is a unique common internal node u of $\mathcal{P}_1$ and $\mathcal{P}_2$ where $\mathcal{P}_1$ and $\mathcal{P}_2$ separate. Assume u belongs to class i and the two leaves represent codewords C_1 and C_2, then the ith component of C_1 and C_2 are not prefix of each other. Therefore, a tree-decodable code is a prefix code.

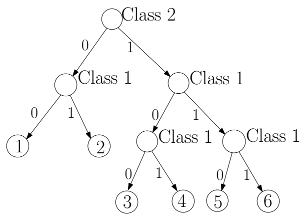

Fig. 3. The decoding tree for the example in Section I.

For the converse, a counterexample is $\mathcal{C} = \{(0,0,B),(1,B,0),(B,1,1)\}$, where B represents a blank. It is trivial to see that $\mathcal{C}$ is a prefix code. If $\mathcal{C}$ has a decoding tree, the root r must belong to some class i_r. If $i_r = 1$, then the first component of all the codewords of $\mathcal{C}$ cannot be blank. Since this is not the case for $\mathcal{C}$, i_r cannot be equal to 1. Similarly, it can be shown that i_r cannot be equal to 2 or 3 either. Thus we conclude that $\mathcal{C}$ does not have a decoding tree. ■

V. Optimal Code Construction

Tree-decodable codes are easy to encode and decode. However, it is not clear whether such codes are optimal. In this section, we show that such codes are optimal with respect to the expected total codeword length, and are also asymptotically optimal in the strongest possible sense. Therefore, for most applications, it is sufficient to consider tree-decodable codes.

For the single-channel case, a Huffman code [1] gives the shortest expected codeword length among all uniquely decodable codes. In the sequel, we present a construction of multichannel tree-decodable codes based on a Huffman code.

For a given information source X, let L_{Huff} be the expected length of a Huffman code. Denote a vector β in the k-dimensional Euclidean space $\Re^k$ by $(\beta_1, \beta_2, ..., \beta_k)$ and consider the following subset of $\Re^k$:

$$\mathcal{H} = \left\{ \beta \in \Re^k : \sum_i \beta_i = L_{\text{Huff}} \text{ and } \forall i, \beta_i \geq 0 \right\}.$$

The following theorem shows that for every $\beta \in \mathcal{H}$, it is possible to construct a tree-decodable code whose expected coding lengths for the individual channels are close to β, while the expected total codeword length is equal to L_{Huff}.

Theorem 8: For any $\beta \in \mathcal{H}$, there exists a q-ary k-channel tree-decodable code $\mathcal{C}$ with expected coding lengths $R_1, R_2, ..., R_k$ for the individual channels such that

$$|R_i - \beta_i| \leq 1$$

for all i and $R = (R_1, R_2, ..., R_k) \in \mathcal{H}$.

Proof: We first choose any q-ary Huffman code for the source X and consider its decoding tree $\mathcal{T}$. We now construct a decoding tree for the required code as follows:

(i) Arbitrarily divide all the internal nodes in $\mathcal{T}$ into k classes.

(ii) Compute the expected coding lengths $R_1, R_2, \ldots, R_k$ for the individual channels. By Theorem 5,

$$\sum_i R_i = \sum_i \sum_{u \in \text{class } i} p_u = L_{\text{Huff}},$$

i.e., $R \in \mathcal{H}$. Moreover,

$$R_1 + R_2 + ... + R_k = \beta_1 + \beta_2 + ... + \beta_k. \qquad (8)$$

(iii) If there exist i such that $R_i - \beta_i > 1$, go to (iv); if there exist i such that $R_i - \beta_i < -1$, go to (v); if for all i, $|R_i - \beta_i| \leq 1$, stop the construction.

(iv) Repeat the following procedure *PP* until $R_i - \beta_i \leq 1$.

PP: By (8), we find a j such that $R_j - \beta_j < 0$. Since $R_i - \beta_i > 1$, we have $R_i > 1 + \beta_i \geq 1 > 0$. Thus there exists at least one internal node u in $\mathcal{T}$ belonging to class i. By reassigning u to class j, R_i is decreased by p_u while R_j is increased by p_u. Since $0 < p_u < 1$, after the reassignment, the new R_i continues to satisfy $R_i - \beta_i \geq -1$ and the new R_j continues to satisfy $|R_j - \beta_j| \leq 1$.

It's easy to see when *PP* stops, we have $|R_i - \beta_i| \leq 1$. Then go back to (iii).
(v) Repeat the following procedure *NP* until $R_i - \beta_i \geq -1$.

NP: By (8), we can find a j such that $R_j - \beta_j > 0$, so that $R_j > \beta_j \geq 0$. Thus there exists at least one internal node u in $\mathcal{T}$ belonging to class j. Then reassign u to class i. After the reassignment, the new R_i continues to satisfy $R_i - \beta_i \leq 1$ and the new R_j continues to satisfy $|R_j - \beta_j| \leq 1$.

It's easy to see when *NP* stops, we have $|R_i - \beta_i| \leq 1$. Then go back to (iii).

We can see that each execution of (iv) (or (v)) requires no more than $|\mathcal{T}|$ (the size of $\mathcal{T}$) iterations of *PP* (or *NP*). Thus we conclude that the construction will stop within $O(k|\mathcal{T}|)$ steps. At the end, $\mathcal{T}$ becomes a decoding tree with the required properties. This completes the proof of the theorem. ■

The k-channel tree-decodable code constructed in Theorem 8 has expected total length equal to L_{Huff}. We now show by contradiction that there does not exist a k-channel tree-decodable code whose expected total codeword length is less than L_{Huff}. Assume such a code exists. Then by reassigning all the internal nodes to class 1, the code becomes a single-channel prefix code with expected codeword length less than L_{Huff}. This contradicts the optimality of a Huffman code.

The next theorem asserts that the tree-decodable code constructed in Theorem 8 not only is optimal among all tree-decodable codes but is in fact optimal among all uniquely decodable codes.

Theorem 9: The tree-decodable code constructed in Theorem 8 is an optimal uniquely decodable code in terms of the expected total codeword length.

Proof: We have shown in Theorem 3 that the set of total codeword lengths of any uniquely decodable code satisfies the Kraft inequality. Since the Kraft inequality is a sufficient condition for the existence of a single-channel prefix code, from any uniquely decodable code, we can construct a single-channel prefix code with expected codeword length the same as the expected total codeword length of the uniquely decodable code. If follows from the optimality of a Huffman code that the expected total codeword length of a uniquely decodable code is lower bounded by L_{Huff}. This proves that the tree-decodable code constructed in Theorem 8 is optimal in terms of the expected total codeword length. ■

The following corollary is a straightforward consequence of Theorem 8.

Corollary 1: For an information source X and any nonnegative real numbers $\alpha_1, \alpha_2, ..., \alpha_k$ satisfying

$$\alpha_1 + \alpha_2 + ... + \alpha_k = H_q(X), \tag{9}$$

there exists a q-ary k-channel tree-decodable code with expected coding lengths $R_1, R_2, ..., R_k$ for the individual channels such that

$$|R_i - \alpha_i| < 1 + \frac{1}{k}$$

for all i and

$$R_1 + R_2 + ... + R_k < H_q(X) + 1.$$

Proof: Let $\delta = L_{\text{Huff}} - H_q(X)$, where $\delta < 1$. For all i, let

$$\beta_i = \alpha_i + \frac{\delta}{k}. \tag{10}$$

Then by (9),

$$\begin{aligned} \sum_i \beta_i &= \sum_i \left(\alpha_i + \frac{\delta}{k} \right) \\ &= H_q(X) + \delta \\ &= L_{\text{Huff}}. \end{aligned}$$

Therefore, $(\beta_1, \beta_2, \ldots, \beta_k) \in \mathcal{H}$. By Theorem 8, there exists a q-ary k-channel tree-decodable code with expected coding lengths $R_1, R_2, ..., R_k$ such that $|R_i - \beta_i| \leq 1$ and $\sum_i R_i = L_{\text{Huff}}$. Finally,

$$\begin{aligned} |R_i - \alpha_i| &\leq |R_i - \beta_i| + |\beta_i - \alpha_i| \\ &\leq 1 + \frac{\delta}{k} \\ &< 1 + \frac{1}{k}, \end{aligned}$$

where the first inequality follows from (10), and

$$R_1 + R_2 + ... + R_k = L_{\text{Huff}} < H_q(X) + 1.$$

The proof is accomplished. ■

Now suppose we use the code construction in Corollary 1 to encode $X_1, X_2, ..., X_n$ which are n i.i.d. copies of X. Let us denote the expected coding length for channel i by R_i^n. Then we have

$$|R_i^n - n\alpha_i| < 1 + \frac{1}{k}.$$

Dividing by n, we obtain

$$\left| \frac{R_i^n}{n} - \alpha_i \right| < \frac{1}{n} + \frac{1}{nk}.$$

As $n \to \infty$, the average expected coding length for channel i, namely the *rate* of the code for channel i, approaches α_i for all i. But of course, as n becomes large, constructing such code becomes very complicated.

Nevertheless, this result indicates that entropy continues to be a fundamental measure of information in multichannel source coding. Furthermore, we see from the proof of Theorem 9 that if a rate tuple $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_k)$ is achievable, then it satisfies

$$\alpha_1 + \alpha_2 + \ldots + \alpha_k \geq H_q(X). \tag{11}$$

Since the tree-decodable code constructed in Theorem 8 can asymptotically achieve any rate tuple α satisfying (11) with equality, it is asymptotically optimal in the strongest possible sense.

VI. Conclusion

In this paper, we have generalized the classical notions of uniquely decodable code, self-punctuating code, and prefix code to zero-error multichannel source coding. In particular, we have studied a class of prefix codes called tree-decodable codes and presented a construction of such codes which shows the tightness of the entropy bound.

References

[1] D. A. Huffman, "A method for the construction of minimum redundancy codes," *Proc. IRE*, 40: 1098-1101, 1952.
[2] Y. Horibe, "An improved bound for weight-balanced tree," *Info. Contr.*, 34: 148-151, 1977.
[3] R. W. Yeung, "Local redundancy and progressive bounds on the redundancy of a Huffman code," *IEEE Trans. Info. Theory*, IT-37: 687-691, 1991.
[4] R. W. Yeung, "On noiseless diagnosis," *IEEE Trans. Info. Theory*, IT-24: 1074-1082, 1994.
[5] L. Song, R. W. Yeung and N. Cai, "Zero-error network coding for acyclic networks," *IEEE Trans. Info. Theory*, IT-49: 3129-3139, 2003.
[6] R. W. Yeung, *Information Theory and Network Coding*, Springer 2008.