The construction of binary Huffman equivalent codes with a greater number of synchronising codewords

Yuh-Ming Huang* and Sheng-Chi Wu

Department of Computer Science and Information Engineering, National Chi Nan University, Nantou, 545, Taiwan
E-mail: ymhuang@csie.ncnu.edu.tw
E-mail: terry.wu.68@gmail.com
*Corresponding author

Abstract: An inherent problem with a Variable-Length Code (VLC) is that even a single bit error can cause a loss of synchronisation, and thus lead to error propagation. Codeword synchronisation has been extensively studied as a means to overcome this drawback and efficiently stop error propagation. In this paper, we first present the sufficient and necessary conditions for the existence of binary Huffman equivalent codes with the shortest, or at most two shortest, synchronising codeword(s) of length \(m + 1\), where \(m (>1)\) is the shortest codeword length. Next, based on the results, we propose a unified approach for constructing each of these binary Huffman equivalent codes with the shortest, or at most two shortest, synchronising codeword(s) of length \(m + 1\), if such a code exists for a given length vector.

Keywords: VLC; variable-length code; Huffman code; Huffman equivalent code; synchronous code; synchronising codeword; ad hoc; ubiquitous computing.

Reference to this paper should be made as follows: Huang, Y-M. and Wu, S-C. (2012) ‘The construction of binary Huffman equivalent codes with a greater number of synchronising codewords’, Int. J. Ad Hoc and Ubiquitous Computing, Vol. 9, No. 1, pp.54–65.

Biographical notes: Yuh-Ming Huang received the BSc in Mathematics from National Tsing Hua University, Hsinchu, Taiwan, in 1987 and the MSc and PhD in Computer Science and Information Engineering from National Taiwan University, Taipei, Taiwan, in 1989 and 1999, respectively. He is now an Assistant Professor with the Department of Computer Science and Information Engineering, National Chi Nan University, Nantou, Taiwan. His current research interests include data compression, error correction coding, joint source/channel coding, video encryption and key agreement in mobile wireless communication.

Sheng-Chi Wu received his BSc Degree in Computer Science and Information Engineering from I-Shou University, Kaohsiung, Taiwan, in 2001, and his MSc Degree in Computer Science and Information Engineering from National Chi Nan University, Nantou, Taiwan, in 2003.

1 Introduction

Variable-Length Code (VLC) is an efficient entropy-coding technology for minimising the total amount of data for image/video information transmission. For instance, Huffman code (Huffman, 1952) has been shown to be optimal in terms of the minimum average codeword length. In addition, there are still some VLCs that have the same average codeword length as a Huffman code, but cannot be constructed by a Huffman algorithm. All of these codes are called “Huffman equivalent codes”. A major problem with a VLC is that if a channel error occurs during transmission, it may lead to the loss of synchronisation for decoding, and the error may propagate and affect the correctness of the next received codewords.

To halt this error propagation, Rudner (1971) defined a synchronising sequence that allows the decoder to resynchronise for a VLC. If a VLC contains at least one synchronising sequence, it is called a statistically synchronisable code, for example the code obtained by Capocelli et al. (1992). The resynchronising ability of this kind of code has also been extensively studied (Wei and Sholtz, 1980; Capocelli et al., 1988).

A synchronous code that has at least one of its codewords as a synchronising sequence belongs to a special class of statistically synchronisable codes. This codeword is also called a synchronising codeword. Ferguson and Rabinowiz (1984) were the first to introduce the definition for a synchronous code. Next, Montgomery and Abrahams (1986) generalised it at the expense of a slight increase in redundancy. Later, Escott and Perkins (1996, 1998) and, Perkins and Escott (1999) provided an algorithm for constructing a binary Huffman equivalent code that contains at least one synchronising codeword of length \(m + 1\), where \(m (>1)\) is the shortest codeword length (the case of \(m = 1\)
was covered in Rudner (1971), if such a code exists for a given length vector.

For the synchronisation problem of a VLC, Takishima et al. (1994) formulated it as a discrete-time Markov chain (Kleinrock, 1975). Through an analysis of error state transition, a good VLC tree structure was suggested, and an algorithm for finding such a code with high synchronisation capability was also proposed. Later, Zhou and Zhang (2002) re-examined the synchronisation capability of a prefix-free code by means of two good measurement criteria, the Mean Error Propagation Length (MEPL) and the Variance of Error Propagation Length (VEPL). They also proposed two algorithms for designing a code with a short MEPL and VEPL. The effect of a binary symmetric channel on the synchronisation behaviour was explored in Zhou et al. (2008). Chabboush and Lamy (2002) proposed another VLC tree structure with good synchronisation behaviour. Higgins et al. (2009) proposed another class of VLCs with good synchronisation properties. Recently, it has been shown that the self-synchronising feature of a synchronising codeword can be integrated with Maximum A-posteriori Probability (MAP) VLC decoding to improve the decoding performance and reduce the complexity (Malinowski et al., 2007; Cao et al., 2007).

In this paper, we first present the sufficient and necessary conditions for the existence of binary Huffman equivalent codes with the shortest, or at most two shortest, synchronising codeword(s) of length \( m + 1 \), where \( m (> 1) \) is the shortest codeword length.

Next, based on the results, we propose a unified approach for constructing each of these binary Huffman equivalent codes with the shortest, or at most two shortest, synchronising codeword(s) of length \( m + 1 \), if such a code exists for a given length vector.

In general, our constructed codes result in a greater number of synchronising codewords than the existing codes in the literature. Moreover, we further show that one of the constructed codes has better synchronisation capability than the existing ones.

2 Preliminaries

Let \( A \) be the set \{0,1\}, and \( A^n \) be the set of all sequences obtained by concatenating \( n \) symbols of \( A \). Let \( A^* = \cup_{i=1}^{\infty} A^n \) be the set of finite sequences of elements of \( A \) and \( A^* = A^* \cup \{ \lambda \} \), where \( \lambda \) is the empty sequence. A sequence with a run of \( r \) ones (resp. zeros) is denoted by 1\(^r\) (resp. 0\(^r\)). A finite subset \( C \) of \( A^* \) is called a binary code, and every \( c \in C \) is called a codeword.

Let \( (n_1, \ldots, n_d) \) be the length vector of code \( C \), where \( n_1 \) and \( M \), respectively, denote the number of codewords of length \( i \) in \( C \) and the maximum length of the codewords in \( C \). In this paper, we suppose that each given length vector \( (n_1, \ldots, n_d) \) satisfies

\[
\sum_{i=1}^{d} n_i 2^{-i} = 1.
\]

That is, the given length vector stands for a Huffman code or a Huffman equivalent code.

Any binary Huffman equivalent code, \( C \), can be represented by a unique binary tree, where each node either has two branches (left-branch denoted as symbol 0 and right-branch denoted as symbol 1) or is a terminal node. The level of a node in the tree is defined by initially letting the root be at level zero. The depth of the tree is defined as the maximum level of nodes in the tree. The path of a node is a string composed of the collection of symbols traversed from the root to that node. A codeword is the path of some terminal node.

In error propagation, an error that occurs in some codeword of the received string causes the codeword to be decoded incorrectly, and then the next codeword(s) (one codeword or many codewords) is (are) also decoded incorrectly. Until some codeword is decoded correctly, the code is resynchronised. The processes of error propagation and resynchronisation in a Huffman code are shown in Figure 1.

![Figure 1](image)

The following definitions and theorem in this section were originally given in Rudner (1971) and Escott and Perkins (1998).

**Definition 1:** Let \( C \) be a binary Huffman equivalent code. We say that \( C \) is synchronous if there is a codeword \( c = c_1c_2\ldots c_r \) in \( C \) satisfying the following two conditions:

- For all \( b = b_1b_2\ldots b_n \) in \( C \) such that \( n > r \) and \( c \) is a substring of \( b \), we have \( c_1c_2\ldots c_r = b_{n+1}b_{n+2}\ldots b_n \), but \( c_1c_2\ldots c_r \neq b_{i+1}b_{i+2}\ldots b_{i+r-1} \) for any \( i \neq n - r + 1 \).

- For any \( j < r \) such that \( c_1c_2\ldots c_j \) appears as a suffix of a codeword, the sequence \( c_{j+1}c_{j+2}\ldots c_r \) must be a sequence of codewords.

If such a codeword \( c \) exists, it is called a synchronising codeword for \( C \).

**Definition 2:** Let \( C \) be a binary Huffman equivalent code with the shortest codeword of length \( m \). Let \( c_1c_2\ldots c_r \) be a synchronising codeword of \( C \) with \( r = m + 1 \). A node, \( N \), of the corresponding binary tree is a \( c \)-node if its path is either \( c_2\ldots c_r \) or \( \tilde{z} \tilde{c}_1\tilde{c}_2\ldots \tilde{c}_k \), where \( \tilde{z} \in A^* \). A node, \( N \), is a \( d \)-node if its path is of the type \( \tilde{z} c_1c_2\ldots c_k \) for some \( k \), \( 2 \leq k < r \). A node, \( N \), that is neither a \( c \)-node nor a \( d \)-node is a 0-node (resp. 1-node) if its path ends in a 0 (resp. 1).
Notice that any shortest codeword must not be a synchronising codeword.

**Theorem 1:** c-nodes are ones that must be terminated (taken as codewords) and d-nodes are nodes that must be extended (cannot be taken as codewords).

**Theorem 2:** Suppose C is a binary equivalent Huffman code with the shortest codeword of length \( m \) \( (m > 1) \). Let \( c_1c_2...c_r \) be a synchronising codeword of C such that \( r = m + 1 \) and \( c_1 = 0 \) (resp. 1). Then, \( c_i = 1 \) (resp. 0) for \( i = 2, ..., r - 1 \). That is, if there exist length-(\( m + 1 \)) synchronising codewords with \( c_1 = 0 \) (resp. 1) for C with the shortest codeword of length \( m \), then they can only be \( 01^{r-2}0 \) or \( 01^{r-1} \) (resp. \( 10^{r-2}1 \) or \( 10^{r-1} \)).

**Proof:** A method that was more straightforward than that of Rudner (1971) was given in Huang and Wu (2003).

Escott and Perkins (1996) pointed out that at most two synchronising codewords \( 01^{r-2}0 \) and \( 01^{r-1} \), or \( 10^{r-2}1 \) and \( 10^{r-1} \) can exist simultaneously in a code, C, if such a code exists. Without loss of generality, we consider the synchronising codewords \( 01^{r-2}0 \) and \( 01^{r-1} \).

### 3 Existence of a code with two synchronising codewords \( 01^{r-2}0 \) and \( 01^{r-1} \) of length \( r \)

**BT:** The corresponding binary tree of code C.

**FBT:** The full binary tree of depth M.

**SFBT:** Any subtree of the FBT.

**\( C_i \):** The number of level \( i \) c-nodes in the BT.

**\( D_i \):** The number of level \( i \) d-nodes in the BT.

**\( 0_i \):** The number of level \( i \) 0-nodes in the BT.

**\( C_F \):** The number of level \( i \) c-nodes in the FBT.

**\( D_F \):** The number of level \( i \) d-nodes in the FBT.

**\( C_0 \):** The number of level \( i \) c-nodes in an SFBT whose root is a 0-node.

**\( C_c \):** The total number of level \( i \) c-nodes in two SFBTs of which the roots are c-nodes; the respective paths of these two c-nodes end, respectively, in a 0 and a 1.

**\( D_0 \):** The number of level \( i \) d-nodes in an SFBT whose root is a 0-node.

**\( D_c \):** The total number of level \( i \) d-nodes in two SFBTs of which the roots are c-nodes; the respective paths of these two c-nodes end, respectively, in a 0 and a 1.

**\( T_0 \):** The number of level \( i \) 0-nodes taken as codewords in the FBT.

**Notes**

- The level of a node in the referenced tree (BT, FBT, or SFBT) is defined by initially letting the root be at level zero.
- The depth of a tree is defined as the maximum level of any node in the tree.

In this section, we derive the sufficient and necessary condition for the co-existence of two length \( r \) synchronising codewords, \( 01^{r-2}0 \) and \( 01^{r-1} \), in a code, C. The sufficient and necessary conditions for the other two cases:

- the existence of a unique length \( r \) synchronising codeword \( 01^{r-1} \)
- the existence of a unique length \( r \) synchronising codeword \( 01^{r-2}0 \), are, respectively, shown in Appendices (A) and (B).

Let C be any binary Huffman equivalent code whose length vector \( (n_1, ..., n_M) \) satisfies \( n_i = 0 \) for \( i < m \) and \( n_m \geq 2 \) for some \( m > 1 \), and with synchronising codewords \( 01^{r-2}0 \) and \( 01^{r-1} \) for \( r = m + 1 \). Then, Lemmas 1–3 hold for such a code.

**Lemma 1:** The number of level \( i \) c-nodes in the BT can be obtained as

\[
C_i = \begin{cases} 
CF_i & \text{for } 1 \leq i < 2m \\
\sum_{k=0}^{\infty} (T0_k \times C0_{i-k}) - \sum_{k=1}^{\infty} \left( \frac{C_k}{2} \times C_{i-k} \right) & \text{for } i \geq 2m, 
\end{cases}
\]

where

\[
\begin{align*}
CF_i &= 0 & \text{for } 1 \leq i < m & \quad (1-1) \\
CF_i &= 2 & \text{for } i = m
\end{align*}
\]

\[
\begin{align*}
CF_i &= 2^{-i(m+1)} \times 2 = 2^{-m} & \text{for } i > m,
\end{align*}
\]

\[
\begin{align*}
C0_i &= 0 & \text{for } 1 \leq i < m & \quad (1-2) \\
C0_i &= 2 & \text{for } i = m
\end{align*}
\]

\[
\begin{align*}
C0_i &= 2^{-i(m+1)} \times 2 = 2^{-m} & \text{for } i > m,
\end{align*}
\]

and

\[
\begin{align*}
C_c &= 0 & \text{for } 1 \leq i < m & \quad (1-3) \\
C_c &= 2 & \text{for } i = m \\
C_c &= 2^{-i(m+1)} \times 2 = 2^{-m+1} & \text{for } i > m.
\end{align*}
\]

**Proof:** (1-1) The number of level \( i \) \( (i < m) \) c-nodes in the FBT is trivially equal to zero. The paths of the two level \( m \) c-nodes in the FBT are \( 1^{r-2}0 \) and \( 1^{r-1} \), respectively. The paths of the level \( i \) \( (i > m) \) c-nodes in the FBT are...
of the form $x_1x_2\ldots x_r01^{r-2}0$ and $x_1x_2\ldots x_r01^{r-1}$, where $x_j (j = 1\ldots i - r) \in \{0, 1\}$. Hence, the number of level $i$ $c$-nodes in the FBT is equal to $2^{(i-r+1)} \times 2 = 2^{i-m}$ for $i > m$ (see Figure 2).

**Figure 2** FBT of depth 6 with two shortest SCs 0110 and 0111, where SC denotes synchronising codeword (see online version for colours)

(1-2) For any SFBT whose root is a 0-node, the number of level $i$ ($i < m$) $c$-nodes in the SFBT is trivially equal to zero. The paths of the two level $m$ $c$-nodes in the SFBT are $1^r0$ and $1^r1$, respectively. The paths of those level $i$ ($i > m$) $c$-nodes in the SFBT are also of the form $x_1x_2\ldots x_r01^{r-2}0$ and $x_1x_2\ldots x_r01^{r-1}$ (see Figure 3).

**Figure 3** Any one SFBT of depth 6 whose root is 0-node when two shortest synchronising codewords, 0110 and 0111, exist in code C (see online version for colours)

(1-3) For any pair of SFBTs whose roots are $c$-nodes, the paths of these two $c$-nodes end, respectively, in a 0 and a 1. The total number of level $i$ $c$-nodes in these two SFBTs is trivially equal to zero for $i < m$. The paths of the two level $m$ $c$-nodes in the former SFBT are $1^r0$ and $1^r1$. However, there exists no $c$-node at level $m$ in the latter SFBT. The paths of the level $i$ ($i > m$) $c$-nodes in either of these two SFBTs are all of the form $x_1x_2\ldots x_r01^{r-2}0$ and $x_1x_2\ldots x_r01^{r-1}$. Hence, the total number of level $i$ $c$-nodes in the pair of SFBTs is equal to $2^{(i-r+1)} \times 2 \times 2 = 2^{i-m+1}$ for $i > m$ (see Figure 4).

From equations (1-1)–(1-3) and $T0_k$, which will be introduced in Lemma 3, $C_i$ can be easily obtained, where $1 \leq i \leq M$.

**Figure 4** Two SFBTs of depth 5 whose roots are $c$-nodes. The paths of these $c$-nodes end, respectively, in a 0 and a 1 (see online version for colours)

**Lemma 2:** The number of level $i$ $d$-nodes in the BT can be obtained as

$$
D_i = D_{F_i} \quad \text{for } 1 \leq i \leq m \text{ and}
$$

$$
D_i = D_{F_i} - \sum_{k=0}^{i-1} (T0_k \times D_{0,k}) - \sum_{k=0}^{i-1} \left( \frac{C_i}{2} \times D_{c,i-k} \right)
$$

for $i > m$,

where

$$
\begin{align*}
D_{F_i} &= 0 \quad \text{for } i = 1 \\
D_{F_i} &= \sum_{k=2}^{i} 2^{i-k} \quad \text{for } 1 < i < m \\
D_{F_i} &= \sum_{k=2}^{i} 2^{i-k} \quad \text{for } i \geq m,
\end{align*}
$$

and

$$
\begin{align*}
D_{0,i} &= 1 \quad \text{for } i = 1 \\
D_{0,i} &= \sum_{k=2}^{i} 2^{i-k} + 1 \quad \text{for } 1 < i < m \\
D_{0,i} &= \sum_{k=2}^{i} 2^{i-k} \quad \text{for } i \geq m,
\end{align*}
$$

and

$$
\begin{align*}
D_{c,i} &= 1 \quad \text{for } i = 1 \\
D_{c,i} &= \sum_{k=2}^{i} 2^{i-k} \times 2 + 1 = \sum_{k=2}^{i} 2^{i-k} \quad \text{for } 1 < i < m \\
D_{c,i} &= \sum_{k=2}^{i} 2^{i-k} \times 2 = \sum_{k=2}^{i} 2^{i-k} \quad \text{for } i \geq m.
\end{align*}
$$

**Proof:** (2-1) The number of level $1$ $d$-nodes in the FBT is trivially equal to zero. The paths of the level $i$ ($1 < i < m$) $d$-nodes in the FBT are of the form $x_1x_2\ldots x_i01^j$, where $j = 1\ldots i - 1$. Hence, the number of level $i$ $d$-nodes in the FBT is equal to
\[ \sum_{i=2}^{m} 2^{-i} \quad \text{for } 1 < i < m. \]

The paths of the level \( i \) \((i \geq m)\) \(d\)-nodes in the FBT are of the form \( x_{i}x_{i+1} \ldots x_{j-1}01^i \), where \( j = 1 \ldots m - 1 \). Hence, the number of level \( i \) \(d\)-nodes in the FBT is equal to

\[ \sum_{i=2}^{m} 2^{-i} \quad \text{for } i \geq m \quad \text{(see Figure 2)}. \]

(2-2) For any SFBT whose root is a 0-node, the number of level \( 1 \) \(d\)-nodes in the SFBT is trivially equal to one. The paths of the level \( i \) \((1 < i < m)\) \(d\)-nodes in the SFBT are of the form \( x_{i}x_{i+1} \ldots x_{j-1}01^i \), where \( j = 1 \ldots i - 1 \) and \( 1 \). Hence, the number of level \( i \) \(d\)-nodes in the SFBT is equal to

\[ 1 + \sum_{i=2}^{m} 2^{i-2} \quad \text{for } 1 < i < m. \]

The paths of the level \( i \) \((i \geq m)\) \(d\)-nodes in the SFBT are of the form \( x_{i}x_{i+1} \ldots x_{j-1}01^i \), where \( j = 1 \ldots m - 1 \). Hence, the number of level \( i \) \(d\)-nodes in the SFBT is equal to

\[ \sum_{i=2}^{m} 2^{i-2} \quad \text{for } i \geq m \quad \text{(see Figure 3)}. \]

(2-3) For any pair of SFBTs whose roots are \(c\)-nodes, the paths of these two roots end, respectively, in a 0 and a 1. There is one level \( 1 \) \(d\)-nodes in the former SFBT, whereas no level \( 1 \) \(d\)-node exists in the latter SFBT. The paths of the level \( i \) \((1 < i < m)\) \(d\)-nodes in the former SFBT are of the forms \( 1 \) and \( x_{i}x_{i+1} \ldots x_{j-1}01^i \), where \( j = 1 \ldots i - 1 \). And, the paths of the level \( i \) \((1 < i < m)\) \(d\)-nodes in the latter SFBT are of the form \( x_{i}x_{i+1} \ldots x_{j-1}01^i \), where \( j = 1 \ldots i - 1 \). Thus, the total number of level \( i \) \(d\)-nodes in the pair of SFBTs is equal to

\[ 1 + \sum_{i=2}^{m} 2^{i-2} \times 2 \quad \text{for } 1 < i < m. \]

The paths of the level \( i \) \((i \geq m)\) \(d\)-nodes in either of these two SFBTs are all of the form \( x_{i}x_{i+1} \ldots x_{j-1}01^i \), where \( j = 1 \ldots m - 1 \). Hence, the total number of level \( i \) \(d\)-nodes in the pair of SFBTs is equal to

\[ \sum_{i=2}^{m} 2^{i-2} \times 2 \quad \text{for } i \geq m \quad \text{(see Figure 4)}. \]

From equations (2-1)-(2-3) and \( T_{0} \), which will be introduced in Lemma 3, \( D_{i} \), can be easily obtained, where \( 1 \leq i \leq M \).

**Lemma 3:** The number of level \( i \) 0-nodes (0) and the number of level \( i \) 0-nodes taken as codewords \( (T_{0}) \) in the BT can be obtained as

\[
\begin{cases}
0_{i} = 2^{i} - D_{i} - 1 & \text{for } 1 \leq i < m \\
0_{i} = 2^{i} - C_{i} - D_{i} & \text{for } i = m \\
0_{i} = \left( D_{i} - C_{i} \right) + \left( 0_{i-1} - T_{0_{i-1}} \right) & \text{for } m < i \leq M
\end{cases}
\]

and

\[
\begin{cases}
T_{0} = 0 & \text{for } 1 \leq i < m \\
T_{0} = n_{i} - C_{i} & \text{for } m \leq i \leq M
\end{cases}
\]

**Proof:** (3-1) Since no \(c\)-node exists and there is just one 1-node at level \( i \) \((1 \leq i < m)\) in the BT, the number of level \( i \) 0-nodes in the BT is equal to \( 2^{i} - D_{i} - 1 \) for \( 1 \leq i < m \). Furthermore, by Corollary 7 of Escott and Perkins (1996), there exists no 1-node of length greater than or equal to \( m \) in the BT. Hence, the number of level \( m \) 0-nodes in the BT is equal to \( 2^{m} - C_{m} - D_{m} \). Moreover, based on Escott and Perkins (1996), extending a level \( i \) 0-node or extending a level \( i \) \(d\)-node with no suffix \( 01^{m-1} \) forms a level-\((i+1)\) 0-node for \( m \leq i \leq M - 1 \). Hence, for \( m < i \leq M \), the number of level \( i \) 0-nodes in the BT is equal to the sum of \( 0_{i-1} - T_{0_{i-1}} \) (which is the number of extended 0-nodes at level \( i - 1 \)) and \( D_{i} - C_{i} \) (which is the number of \( d\)-nodes at level \( i - 1 \) with no suffix \( 01^{m-1} \)).

(3-2) The number of level \( i \) 0-nodes taken as codewords in the FBT is trivially equal to zero for \( 1 \leq i < m \). On the other hand, no 1-node exists at level \( i \), \( m \leq i \leq M \) in the BT. Thus, only 0-nodes and \(c\)-nodes can be taken as codewords, and all \(c\)-nodes must be taken as codewords in the BT. Hence, the number of level \( i \) 0-nodes taken as codewords in the BT is equal to \( n_{i} - C_{i} \) for \( m \leq i \leq M \).

Next, through the computations in Lemma 1, Lemmas 2 and 3, we obtain the main result of this section.

**Theorem 3:** For the length vector \((n_{1}, \ldots, n_{d})\), where \( n_{i} = 0 \) for \( i < m \) and \( n_{m} \geq 2 \) for some \( m > 1 \), there exists a binary Huffman equivalent code, \( C \), that contains two synchronising codewords, \( 01^{m-2}0 \) and \( 01^{m-1} \), with \( r = m+1 \), if and only if \( C_{i} \leq n_{i} \leq C_{i} + 0 \), for \( m \leq i \leq M \).

**Proof:** Since all \(c\)-nodes must be taken as codewords in the BT, \( n_{i} \) must be greater than or equal to \( C_{i} \); otherwise, some \(c\)-nodes will be extended. On the other hand, because only the 0-nodes and \(c\)-nodes can be taken as codewords in this BT, \( n_{i} \) must be less than or equal to \( C_{i} + 0 \); otherwise, some \(d\)-nodes will be terminated. Therefore, \( C_{i} \leq n_{i} \leq C_{i} + 0 \), for \( m \leq i \leq M \) and the sufficient condition part of the theorem is proved.

Since Lemmas 1–3 hold for such a code, \( C \), the inequalities, \( C_{i} \leq n_{i} \leq C_{i} + 0 \), for \( m \leq i \leq M \), assert the existence of the code (i.e., the BT) and the necessary condition part of the theorem is proved.

4 A unified algorithm for constructing a binary Huffman equivalent code with the shortest, or at most two shortest, synchronising codeword(s) of length \( r \)

In Escott and Perkins (1998), the authors pointed out that when \( 01^{r-2}0 \) is the shortest synchronising codeword, a better code (in terms of the synchronising capability) can sometimes be generated by extending the 0-nodes rather
than always extending the 1-nodes. Here, we give the condition under which this criterion can be used, and obtain a better code.

**Theorem 4:** If all of these equations,

\[
\begin{align*}
(E_1): & \quad C_i < n_i < C_i + 0_1 + 1_i, \quad 0 > 0 \quad \text{and} \quad 1 > 0; \\
(E_2): & \quad n_i + 1 < C_i + 0_1 + 1_i; \\
& \vdots \\
(E_m): & \quad n_i + m - 1 < C_i + 0_1 + 1_i; \\
(E_m+1): & \quad C_i < n_i + m;
\end{align*}
\]

hold for some \( i \), where \( m \leq i \leq M - m \), then during the construction of code \( C \), there will exist at least one pair of terminated 0-node and extended 1-node at level \( i \) in the corresponding binary tree. For this moment, if we swap them (i.e., extending the 0-node and terminating the 1-node), then we can also finally obtain a binary equivalent code, \( C' \), with one synchronising codeword, 01\(^{r-1}\), for the same length vector, which is sometimes better than \( C \), with a greater number of synchronising codewords.

Proof \((E_1):\) This implies that at least one 0-node can be terminated \((T_0 > 0, \text{i.e.,} \, n_i > C_i \quad \text{and} \quad 0 > 0)\) and at least one 1-node can be extended \((T_1 < 1, \text{i.e.,} \, n_i - C_i - 0_1 - 1_i < 0)\).

\((E_2-E_m):\) On the basis of Lemma 7 of Escott and Perkins (1998), extending a level \( i \) 0-node forms 0-nodes of every level, \((i + l)\) for \( l = 1 \ldots r - 2; \) d-nodes of every level, \((i + l)\) for \( l = 1 \ldots r - 2; \) a level \((i + r - 1)\) c-node; a level \((i + r - 2)\) 1-node, whereas extending a level \( i \) 1-node forms a level-(\( i + 1 \)) 0-node and a level-(\( i + 1 \)) 1-node. Since any d-node must be extended, only when at least one extended 1-node \((T_1 < 1, \text{i.e.,} \, T_1 = n_i - C_i - 0_1 - 1_i)\) exists for each of the following levels, \((i + h)\) for \( 1 \leq h \leq m - 1 \), is it possible to swap them.

\((E_m+1):\) Since the number of level-(\( i + m \)) c-nodes will increase by one after swapping, \( n_i + m \) must be greater than \( C_i + m \); otherwise, Theorem B.1 will no longer hold.

After swapping, the notations in equation (5) need to be updated to fulfil Theorem B.1, which was originally derived based on the constructing algorithm of Escott and Perkins (1998).

\[
\begin{align*}
1_{i+1} &= 1_{i-1}; \\
1_{i+2} &= 1_{i+2}; \\
& \vdots \\
1_{i+m-1} &= 1_{i+m-1}; \\
0_{i+m} &= 0_{i+m} - 1 \quad \text{and} \quad C_i &= C_i + 1;
\end{align*}
\]

Next, based on the sufficient and necessary conditions (derived in Section 3 and appendices) and Theorem 4, we propose a unified construction algorithm guaranteed to generate a binary Huffman equivalent code with the shortest, or at most two shortest, synchronising codeword(s) of length \( r \), if such a code exists for a given length vector. Furthermore, the number of synchronising codewords of the constructed code is greater than or equal to that of any that exists in the literature.

**Algorithm 1**

**Input:** A length vector \((n_1, \ldots, n_M)\) with \( n_i = 0 \) for \( 1 \leq i < m \)
and \( n_m \geq 1 \) for some \( m > 1 \).

**Output:** A synchronising binary Huffman equivalent code, \( C \).

**Step 1:** Let \( m \) be the smallest integer satisfying \( n_m \neq 0 \), and let \( M \) be the largest integer satisfying \( n_M \neq 0 \). Put \( r = m+1 \).

**Step 2:** If \( n_m = 1 \), then go to Step 3.

If both synchronising codewords, 01\(^{r-2}\)0 and 01\(^{r-1}\), can exist simultaneously in code \( C \) (tested by Theorem 3), then 01\(^{r-2}\)0 and 01\(^{r-1}\) are selected, and go to Step 5.

**Step 3:** If the synchronising codeword, 01\(^{r-1}\), can exist in code \( C \) (tested by Theorem A.1), then 01\(^{r-1}\) is selected, and go to Step 5.

**Step 4:** If the synchronising codeword, 01\(^{r-2}\)0, can exist in code \( C \) (tested by Theorem B.1), then 01\(^{r-2}\)0 is selected, and go to Step 6.

else Return “There exists no binary Huffman equivalent code \( C \) with at least one synchronising codeword of length \( r \) for this length vector”.

**Step 5:** Repeat for each length \( i \), where \( 1 \leq i \leq M \) \{
Terminate all level \( i \) c-nodes.
Extend all level \( i \) d-nodes.
If \( i \geq m \), then terminate any 0-nodes as required, and extend the remaining 0-nodes.
else extend the remaining 0-nodes and 1-nodes.
\} Return \( C \).

**Step 6:** Repeat for each length \( i \), where \( 1 \leq i \leq M \) \{
Terminate all level \( i \) c-nodes.
Extend all level \( i \) d-nodes.
If \( m \leq i \leq M - m \) then \{
Swapno = 0
Repeat \{
If equation (4) holds & (swapno < 1), then \{terminate any one 0-node and extend any one 0-node simultaneously, swapno=swapno+1, and update equation (5)\},
else if any one 0-node is available then terminate any one 0-node.
else terminate any one 1-node.
\} Until \((n_i - C_i)\) nodes are terminated
\} until \((M - m) < i \leq M \) then terminate any 0-nodes whenever possible, otherwise 1-nodes as required.
Extend the remaining 0-nodes and 1-nodes.
\} Return \( C \).
5 Examples

Example 1: For the length vector \((0, 0, 2, 7, 7, 5, 1, 1, 1, 2)\), a corresponding binary Huffman equivalent code, \(C\), with synchronising codeword 0110 can be generated by using Algorithm 1.

Step 1: \(m = 3, M = 10,\) and \(r = 4\).

Step 2: From Table 1, we can ensure that no Huffman equivalent code exists for the given length vector with two synchronising codewords, 0110 and 0111, through the test of Theorem 3.

Step 3: From Table 2, we can further ensure that there is also no Huffman equivalent code for the given length vector, even with only one synchronising codeword, 0111, through the test of Theorem A.1.

Step 4: From Table 3, we know that one Huffman equivalent code exists for the given length vector with a synchronising codeword, 0110, through the test of Theorem B.1. Then, go to Step 6.

Step 6: In this step, equation (4) will hold for the cases of \(i = 3\) and \(i = 7\).

Table 1
Test of Theorem 3

<table>
<thead>
<tr>
<th>Level (i)</th>
<th>(n_i)</th>
<th>(C_i)</th>
<th>(0_i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>6</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 2
Test of Theorem A.1

<table>
<thead>
<tr>
<th>Level (i)</th>
<th>(n_i)</th>
<th>(C_i)</th>
<th>(0_i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>2</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 3
Test of Theorem B.1

<table>
<thead>
<tr>
<th>Level (i)</th>
<th>(n_i)</th>
<th>(C_i)</th>
<th>(0_i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4
Comparisons of MEPLs and VEPLs of Huffman equivalent codes for English alphabet source

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>0.1278</td>
<td>001</td>
<td>001</td>
<td>001</td>
<td>110</td>
<td>110</td>
</tr>
<tr>
<td>T</td>
<td>0.0855</td>
<td>101</td>
<td>101</td>
<td>010</td>
<td>100</td>
<td>111</td>
</tr>
<tr>
<td>O</td>
<td>0.0804</td>
<td>0001</td>
<td>1001*</td>
<td>0001</td>
<td>0110*</td>
<td>0110*</td>
</tr>
<tr>
<td>A</td>
<td>0.0778</td>
<td>1001*</td>
<td>1101</td>
<td>0110</td>
<td>0000</td>
<td>0000</td>
</tr>
<tr>
<td>N</td>
<td>0.0686</td>
<td>0101</td>
<td>0001</td>
<td>0111</td>
<td>0010</td>
<td>0010</td>
</tr>
<tr>
<td>I</td>
<td>0.0667</td>
<td>1101</td>
<td>0101</td>
<td>1010</td>
<td>0100</td>
<td>0100</td>
</tr>
<tr>
<td>R</td>
<td>0.0651</td>
<td>0111</td>
<td>0111</td>
<td>1011</td>
<td>0111</td>
<td>0111</td>
</tr>
<tr>
<td>S</td>
<td>0.0622</td>
<td>1111*</td>
<td>1111</td>
<td>1100</td>
<td>1010</td>
<td>1000</td>
</tr>
<tr>
<td>H</td>
<td>0.0595</td>
<td>0000</td>
<td>0000</td>
<td>1111</td>
<td>1110</td>
<td>1010</td>
</tr>
<tr>
<td>D</td>
<td>0.0404</td>
<td>10001*</td>
<td>01001*</td>
<td>000001</td>
<td>00110*</td>
<td>00110*</td>
</tr>
</tbody>
</table>
Table 4: Comparisons of MEPLs and VEPLs of Huffman equivalent codes for English alphabet source (continued)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>0.0372</td>
<td>01001*</td>
<td>11001*</td>
<td>10010</td>
<td>10110*</td>
<td>10110*</td>
</tr>
<tr>
<td>U</td>
<td>0.0308</td>
<td>11001*</td>
<td>11001</td>
<td>10011</td>
<td>00010</td>
<td>00010</td>
</tr>
<tr>
<td>C</td>
<td>0.0296</td>
<td>01101*</td>
<td>01101</td>
<td>11010</td>
<td>01010</td>
<td>01111</td>
</tr>
<tr>
<td>M</td>
<td>0.0288</td>
<td>11101</td>
<td>11001</td>
<td>11101</td>
<td>10111</td>
<td>10100</td>
</tr>
<tr>
<td>P</td>
<td>0.0223</td>
<td>10000</td>
<td>11000</td>
<td>11010</td>
<td>11110</td>
<td>10010</td>
</tr>
<tr>
<td>F</td>
<td>0.0197</td>
<td>01000</td>
<td>11000</td>
<td>11101</td>
<td>11111</td>
<td>10111*</td>
</tr>
<tr>
<td>Y</td>
<td>0.0196</td>
<td>110001*</td>
<td>110001*</td>
<td>100001*</td>
<td>000110*</td>
<td>000110*</td>
</tr>
<tr>
<td>W</td>
<td>0.0176</td>
<td>011001*</td>
<td>011001*</td>
<td>100010</td>
<td>010110*</td>
<td>010110*</td>
</tr>
<tr>
<td>G</td>
<td>0.0174</td>
<td>1110001*</td>
<td>1110001*</td>
<td>1000001*</td>
<td>0011110*</td>
<td>1001110</td>
</tr>
<tr>
<td>B</td>
<td>0.0141</td>
<td>1100001*</td>
<td>1100001*</td>
<td>100010</td>
<td>0011110</td>
<td>1001110</td>
</tr>
<tr>
<td>V</td>
<td>0.0112</td>
<td>0110001*</td>
<td>0110001*</td>
<td>10010</td>
<td>011110</td>
<td>1001111</td>
</tr>
<tr>
<td>K</td>
<td>0.0074</td>
<td>11100001*</td>
<td>11100001*</td>
<td>100001*</td>
<td>00111100*</td>
<td>10011100</td>
</tr>
<tr>
<td>J</td>
<td>0.0051</td>
<td>111000001*</td>
<td>111000001*</td>
<td>10000001*</td>
<td>001111110*</td>
<td>100111100</td>
</tr>
<tr>
<td>X</td>
<td>0.0027</td>
<td>1110000001*</td>
<td>1110000001*</td>
<td>100000001*</td>
<td>0011111110*</td>
<td>1001111100</td>
</tr>
<tr>
<td>Z</td>
<td>0.0017</td>
<td>11100000001*</td>
<td>11100000001*</td>
<td>1000000001*</td>
<td>00111111110*</td>
<td>10011111010</td>
</tr>
<tr>
<td>Q</td>
<td>0.0008</td>
<td>111000000001*</td>
<td>111000000001*</td>
<td>10000000001*</td>
<td>0011111111110</td>
<td>1001111101111</td>
</tr>
<tr>
<td>MEPL</td>
<td>1.9030</td>
<td>1.8483</td>
<td>3.3998</td>
<td>2.0522</td>
<td>1.9260</td>
<td>1.5225</td>
</tr>
<tr>
<td>VEPL</td>
<td>1.3634</td>
<td>1.2294</td>
<td>7.2986</td>
<td>2.0672</td>
<td>1.5225</td>
<td></td>
</tr>
</tbody>
</table>

*Denotes that the codeword is a synchronising codeword.

Example 2: For the length vector (0, 2, 3, 2), a binary Huffman equivalent code, C, with two synchronising codewords, 010 and 011, as shown in Table 5, can be generated by using Algorithm 1. However, only one synchronising codeword, 101, exists in the code obtained by using either the Fixed Order method or the Max Gain method of Zhou and Zhang (2002). The MEPL and VEPL values of the code generated from Algorithm 1 are smaller.

Table 5: Comparisons of MEPLs and VEPLs of Huffman equivalent codes for length vector (0, 2, 3, 2)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.25 01</td>
<td>01</td>
<td>10</td>
</tr>
<tr>
<td>B</td>
<td>0.25 11</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>C</td>
<td>0.125 101*</td>
<td>101*</td>
<td>010*</td>
</tr>
<tr>
<td>D</td>
<td>0.125 001</td>
<td>001</td>
<td>011*</td>
</tr>
<tr>
<td>E</td>
<td>0.125 000</td>
<td>000</td>
<td>000</td>
</tr>
<tr>
<td>F</td>
<td>0.0625 1001*</td>
<td>1001*</td>
<td>0010*</td>
</tr>
<tr>
<td>G</td>
<td>0.0625 1000*</td>
<td>1000*</td>
<td>0011*</td>
</tr>
<tr>
<td>MEPL</td>
<td>1.9252</td>
<td>1.9252</td>
<td>1.8163</td>
</tr>
<tr>
<td>VEPL</td>
<td>1.1318</td>
<td>1.1318</td>
<td>0.9637</td>
</tr>
</tbody>
</table>

6 Conclusion and discussion

In this paper, we elaborately derived the sufficient and necessary conditions for the existence of binary Huffman equivalent codes with the shortest, or at most two shortest, synchronising codeword(s) of length r, and proposed a unified approach for constructing these codes. We also showed that one code constructed by the proposed algorithm had better synchronisation capability than the existing ones.
The results of Zhou and Zhang (2002) are significant, but the synchronisation capability of the code constructed by the construction algorithm of Zhou and Zhang (2002) is not always the best. A method for combining the ideas of Zhou and Zhang (2002) and the synchronising codeword technique to design a much better code deserves further investigation.

References


Appendices

In Appendices (A) and (B), since there are no two c-nodes with the same parent, the abbreviated notations, $C_i$ and $D_i$, are redefined as follows:

- $C_i$: The number of level $i$ c-nodes in an SFBT whose root is a c-node.
- $D_i$: The number of level $i$ d-nodes in an SFBT whose root is a c-node.

For the following lemmas and theorems, we will omit portions of the proofs because these proofs are similar to those of Section 3.

(A) Existence of a code with a unique synchronising codeword $01^{r-1}$ of length $r$

Let $C$ be any binary Huffman equivalent code whose length vector $(n_1, \ldots, n_m)$ satisfies $n_i = 0$ for $i < m$ and $n_m \geq 1$ for some $m > 1$, and with only one synchronising codeword $01^{r-1}$ of length $r$, where $r = m + 1$. Then, Lemmas A.1–A.3 hold for such a code.

**Lemma A.1:** The number of level $i$ c-nodes in the BT can be obtained as

$$C_i = \begin{cases} C_{i-1} & \text{for } 1 \leq i < 2m \\ C_{i-1} - \sum_{k=m}^{i-1} (T_0 \times C_{i-1-k}) - \sum_{k=m}^{i-1} (C_{i-1} \times C_{i-1-k}) & \text{for } i \geq 2m, \end{cases}$$

where

$$C_i = \begin{cases} 0 & \text{for } 1 \leq i < m \\ 1 & \text{for } i = m \\ 2^{i-(m+1)} & \text{for } i > m, \end{cases}$$

and
The construction of binary Huffman equivalent codes with a greater number of synchronising codewords

\[ C_i = 0 \quad \text{for} \quad 1 \leq i \leq m \]
\[ C_i = 2^{-\left(\text{m+1}\right)} \quad \text{for} \quad i > m. \]

(A1-3) \[ \sum_{i=2}^{m} 2^{-i} \quad \text{for} \quad i \geq m. \]

Proof: (A1-3) The number of level \( i \) c-nodes in an SFBT whose root is a c-node is trivially equal to zero for \( 1 \leq i \leq m \). The path of the level \( i \) (\( i > m \)) c-nodes in the SFBT is of the form \( x_1 x_2 \ldots x_{j-1} 01^{i-1} \), where \( j = 1 \ldots i-1 \). Hence, the number of level \( i \) c-nodes in the SFBT is equal to \( 2^{-(\text{m+1})} \) for \( i > m \). \hfill \Box

Lemma A.2: The number of level \( i \) d-nodes in the BT can be obtained as

\[ D_i = DF_i \quad \text{for} \quad 1 \leq i \leq m \]
\[ D_i = DF_i - \sum_{k=m}^{i-1} (T0_k \times D0_{i-k}) - \sum_{k=m}^{i-1} (C_k \times DC_{i-k}) \quad \text{for} \quad i > m, \]

where

\[
\begin{align*}
DF_i &= 0 \quad \text{for} \quad i = 1 \\
DF_i &= \sum_{k=2}^{i-1} 2^{-i-k} \quad \text{for} \quad 1 < i < m \\
DF_i &= \sum_{k=2}^{i} 2^{-i-k} \quad \text{for} \quad i \geq m.
\end{align*}
\]

(A2-1)

\[ D0_i = 1 \quad \text{for} \quad i = 1 \]
\[ D0_i = \sum_{k=2}^{i-1} 2^{-i-k} + 1 \quad \text{for} \quad 1 < i < m \]
\[ D0_i = \sum_{k=2}^{i} 2^{-i-k} \quad \text{for} \quad i \geq m, \]

(A2-2)

and

\[ DC_i = 0 \quad \text{for} \quad i = 1 \]
\[ DC_i = \sum_{k=2}^{i-1} 2^{-i-k} \quad \text{for} \quad 1 < i < m \]
\[ DC_i = \sum_{k=2}^{i} 2^{-i-k} \quad \text{for} \quad i \geq m. \]

(A2-3)

Proof: (A2-3) The number of level \( 1 \) d-nodes in an SFBT whose root is a c-node is trivially equal to zero. The paths of those level \( i \) (\( 1 < i < m \)) d-nodes in the SFBT are of the form \( x_1 x_2 \ldots x_{j-1} 01^{j-1} \), where \( j = 1 \ldots i-1 \). Hence, the number of level \( i \) d-nodes in the SFBT is equal to

The paths of the level \( i \) (\( i \geq m \)) d-nodes in the SFBT are of the form \( x_1 x_2 \ldots x_{j-1} 01^{i-1} \), where \( j = 1 \ldots m-1 \). Hence, the number of level \( i \) d-nodes in the SFBT is equal to

\[ \sum_{i=2}^{m} 2^{-i} \quad \text{for} \quad 1 < i \leq m. \]

\[ \sum_{i=2}^{m} 2^{-i} \quad \text{for} \quad i \geq m. \]

Lemma A.3: The number of level \( i \) 0-nodes (0) and the number of level \( i \) 0-nodes taken as codewords (T0) in the BT can be obtained as

\[ 0 \quad = 2^i \quad - D_i - 1 \quad \text{for} \quad 1 \leq i < m \]
\[ 0 \quad = 2^i \quad - C_i - D_i \quad \text{for} \quad i = m \]
\[ 0 \quad = D_{i+1} + (0_{i-1} - T0_{i-1}) \quad \text{for} \quad m < i \leq M \]

(A3-1)

\[ T0_0 = 0 \quad \text{for} \quad 1 \leq i < m \]
\[ T0_i = n_i - C_i \quad \text{for} \quad m \leq i \leq M. \]

(A3-2)

Proof: (A3-1) On the basis of Escott and Perkins (1998), extending a level 0-node or extending a level \( i \) d-node must form a level-(\( i + 1 \)) 0-node for \( m \leq i \leq M-1 \). Hence, for \( m < i \leq M \), the number of level \( i \) 0-nodes in the BT is equal to the sum of \( 0_{i-1} - T0_{i-1} \) (which is the number of extended 0-nodes at level \( i-1 \)) and \( D_{i-1} \) (which is the number of d-nodes at level \( i-1 \)). \hfill \Box

Next, through the computations in Lemma A.1, Lemma A.2 and Lemma A.3, we have the following theorem.

Theorem A.1: For the length vector \( (n_1, \ldots, n_M) \), where \( n_1 = 0 \) for \( i < m \) and \( n_1 \geq 1 \) for some \( m > 1 \), there exists a binary Huffman equivalent code, \( C \), that contains only one synchronising codeword, \( 01^{r-1} \), of length \( r \), with \( r = m + 1 \), if and only if \( C_i \leq n_i \leq C_i + 0 \), for \( m \leq i \leq M \) and the sufficient condition part of the theorem is proved.

Proof: Since all c-nodes must be taken as codewords in the BT, \( n_i \) must be greater than or equal to \( C_i \); otherwise, some c-nodes will be extended. On the other hand, because only the 0-nodes and c-nodes can be taken as codewords in this BT, \( n_i \) must be less than or equal to \( C_i + 0 \); otherwise, some d-nodes will be terminated. Therefore, \( C_i \leq n_i \leq C_i + 0 \), for \( m \leq i \leq M \) and the sufficient condition part of the theorem is proved.

Since Lemmas A.1–A.3 hold for such a code, \( C \), the inequalities, \( C_i \leq n_i \leq C_i + 0 \), for \( m \leq i \leq M \), assert the existence of the code (i.e., the BT) and the necessary condition part of the theorem is proved. \hfill \Box

(B) Existence of a code with a unique synchronising codeword \( 01^{r-2}0 \) of length \( r \)

\[ C1_i: \] The number of level \( i \) c-nodes in an SFBT whose root is a 1-node.
\[ D1_i: \] The number of level \( i \) d-nodes in an SFBT whose root is a 1-node.
\[ 1_j: \] The number of level \( 1 \) 1-nodes in the BT.
\[ T1_i: \] The number of level \( i \) 1-nodes taken as codewords in the FBT.
Let \((n_1, \ldots, n_d)\) be the length vector of any binary Huffman equivalent code, \(C\), with only one synchronising codeword \(01^{m-1}0\), of length \(r = m + 1\), where \(n_i = 0\) for \(i < m\) and \(n_m \geq 1\) for some \(m > 1\). Suppose this code is constructed by using Algorithm 2 of Escott and Perkins (1998), and always terminates 0-nodes (alternatively extends 1-nodes) whenever possible. Then, Lemmas B.1–B.4 hold for such a code.

**Lemma B.1:** The number of level \(i\) \(c\)-nodes in the BT can be obtained as

\[
\begin{align*}
C_i &= CF_i \quad \text{for } 1 \leq i < 2m \\
C_i &= CF_i - \sum_{k=0}^{m-1} (T_0 \times CC_i_{-k}) \\
&\qquad - \sum_{k=0}^{m} (C_i \times Cc_{i-1-k}) - \sum_{k=0}^{m} (T_1 \times CL_{i-1-k}) \\
&\quad \text{for } i \geq 2m,
\end{align*}
\]

where

\[
\begin{align*}
CF_i &= 0 \quad \text{for } 1 \leq i < m \\
CF_i &= 1 \quad \text{for } i = m \\
CF_i &= 2^{i-(m+1)} \quad \text{for } i > m, \\
C0_i &= 0 \quad \text{for } 1 \leq i < m \\
C0_i &= 1 \quad \text{for } i = m \\
C0_i &= 2^{i-(m+1)} \quad \text{for } i > m, \\
CL_i &= 0 \quad \text{for } 1 \leq i < m \\
CL_i &= 1 \quad \text{for } i = m \\
CL_i &= 2^{i-(m+1)} \quad \text{for } i > m, \\
\end{align*}
\]

and

\[
\begin{align*}
Cc_i &= 0 \quad \text{for } 1 \leq i < m \\
Cc_i &= 1 \quad \text{for } i = m \\
Cc_i &= 2^{i-(m+1)} \quad \text{for } i > m.
\end{align*}
\]

**Proof:** (B1-4) The number of level \(i\) \(c\)-nodes in an SFBT whose root is a \(c\)-node is trivially equal to one. The paths of the level \(i \quad (1 \leq i < m)\) \(d\)-nodes in the SFBT are of the form \(1^i\) and \(x_1x_2\ldots x_{i-j},01^{j-1}\), where \(j = 1\ldots i - 1\). Hence, the number of level \(i\) \(d\)-nodes in the SFBT is equal to \(2^{i-(m+1)}\) for \(i > m\).

**Lemma B.2:** The number of level \(i\) \(d\)-nodes in the BT can be obtained as

\[
\begin{align*}
D_i &= DF_i \quad \text{for } 1 \leq i \leq m \\
D_i &= DF_i - \sum_{k=0}^{m-1} (T_0 \times DD_{i-1-k}) - \sum_{k=0}^{m} (C_i \times DC_{i-1-k}) \\
&\qquad - \sum_{k=0}^{m} (T_1 \times DL_{i-1-k}) \quad \text{for } i > m,
\end{align*}
\]

where

\[
\begin{align*}
DF_i &= 0 \quad \text{for } i=1 \\
DF_i &= \sum_{k=2}^{i} 2^{i-k} \quad \text{for } 1 < i < m \quad \text{(B2-1)} \\
DF_i &= \sum_{k=2}^{m} 2^{i-k} \quad \text{for } i \geq m,
\end{align*}
\]

\[
\begin{align*}
D0_i &= 1 \quad \text{for } i=1 \\
D0_i &= \sum_{k=2}^{i} 2^{i-k} + 1 \quad \text{for } 1 < i < m \quad \text{(B2-2)} \\
D0_i &= \sum_{k=2}^{m} 2^{i-k} \quad \text{for } i \geq m,
\end{align*}
\]

\[
\begin{align*}
D1_i &= 0 \quad \text{for } i=1 \\
D1_i &= \sum_{k=2}^{i} 2^{i-k} \quad \text{for } 1 < i < m \quad \text{(B2-3)} \\
D1_i &= \sum_{k=2}^{m} 2^{i-k} \quad \text{for } i \geq m,
\end{align*}
\]

and

\[
\begin{align*}
Dc_i &= 1 \quad \text{for } i=1 \\
Dc_i &= \sum_{k=2}^{i} 2^{i-k} + 1 \quad \text{for } 1 < i < m \quad \text{(B2-4)} \\
Dc_i &= \sum_{k=2}^{m} 2^{i-k} \quad \text{for } i \geq m.
\end{align*}
\]

**Proof:** (B2-4) The number of level \(i\) \(d\)-nodes in an SFBT whose root is a \(c\)-node is trivially equal to one. The paths of the level \(i \quad (1 \leq i < m)\) \(d\)-nodes in the SFBT are of the form \(1^i\) and \(x_1x_2\ldots x_{i-j},01^{j-1}\), where \(j = 1\ldots i - 1\). Hence, the number of level \(i\) \(d\)-nodes in the SFBT is equal to

\[
1 + \sum_{k=2}^{i} 2^{i-k} \quad \text{for } 1 < i < m.
\]

The paths of the level \(i \quad (i \geq m)\) \(d\)-nodes in the SFBT are of the form \(x_1x_2\ldots x_{i-j},01^{j-1}\), where \(j = 1\ldots m - 1\). Hence, the number of level \(i\) \(d\)-nodes in the SFBT is equal to

\[
\sum_{k=2}^{m} 2^{i-k} \quad \text{for } i \geq m.
\]

**Lemma B.3:** The number of level \(i\) \(1\)-nodes (\(1_i\)) and the number of level \(i\) \(1\)-nodes taken as codewords \((T_1)\) in the BT can be obtained as

\[
\begin{align*}
1_i &= 1 \quad \text{for } 1 \leq i \leq m \\
1_i &= C_i + (1_{i-1} - T1_{i-1}) \quad \text{for } m < i \leq M \quad \text{(B3-1)}
\end{align*}
\]

and

\[
T_1 = \begin{cases}
0 & \text{for } 1 \leq i < m \\
0 & \text{if } C_i = C_i + 0, \\
0 - C_i & \text{if } C_i + 0 < n_i < C_i + 0 + 1, \\
-0 & \text{for } m \leq i < M.
\end{cases}
\]

(B3-2)

Proof: (B3-1) There exists only one 1-node whose path is 1' at each level \(i \leq i \leq m\) in the BT. On the basis of Escott and Perkins (1998), extending a level 1-node or extending a level \(i\) \(-\)node with suffix 01\(^{i-2}\) forms a level-\((i + 1)\) 1-node for \(m \leq i \leq M - 1\). Hence, the number of level 1 1-nodes is equal to the sum of \(1\)-nodes at level \(i - 1\) and \(C_i\) (which is the number of level-\((i - 1)\) \(-\)nodes with suffix 01\(^{i-2}\)) for \(m \leq i \leq M\).

(B3-2) There exists no terminal node at each level \(i \leq i \leq m\) in the BT. Hence, the number of level \(i\) 1-nodes taken as codewords in the BT is equal to zero for \(i \leq i \leq m\). In addition, all of the c-nodes must be taken as codewords in the BT. In the constructing algorithm of Escott and Perkins (1998), 0-nodes are always taken as codewords first whenever possible. Hence, for \(m \leq i \leq M\), the number of level \(i\) 0-nodes taken as codewords is equal to \(n_i - C_i\) if \(C_i = n_i \leq C_i + 0\), and 0 if \(n_i > C_i + 0\).

Next, through the computations in Lemma B.1, Lemma B.2, Lemma B.3 and Lemma B.4, we have the following corollary.

**Corollary B.1:** For the length vector \((n_1, \ldots, n_d)\), where \(n_i = 0\) for \(i < m\) and \(n_m \geq 1\) for some \(m > 1\), there exists a binary Huffman equivalent code, \(C\), that is constructed by using algorithm 2 of Escott and Perkins (1998), and always terminates 0-nodes whenever possible and contains only one synchronising codeword, 01\(^{i-2}0\), of length \(r = m + 1\), if and only if \(C_i = n_i \leq C_i + 0 + 1\) for \(m \leq i \leq M\).

Proof: Since all c-nodes must be taken as codewords in the BT, \(n_i\) must be greater than or equal to \(C_i\); otherwise, some c-nodes will be extended. On the other hand, because only the 0-nodes, 1-nodes and c-nodes can be taken as codewords in this BT, \(n_i\) must be less than or equal to \(C_i + 0 + 1\); otherwise, some \(-\)nodes will be terminated. Therefore, \(C_i = n_i \leq C_i + 0 + 1\), for \(m \leq i \leq M\) and the sufficient condition part of the corollary is proved.

Since Lemmas B.1–B.4 hold for such a code, \(C\), the inequalities, \(C_i \leq n_i \leq C_i + 0 + 1\), for \(m \leq i \leq M\), assert the existence of the code (i.e., the BT) and the necessary condition part of the corollary is proved.

Now, we have the following theorem.

**Theorem B.1:** For the length vector \((n_1, \ldots, n_d)\), where \(n_i = 0\) for \(i < m\) and \(n_m \geq 1\) for some \(m > 1\), there exists a binary Huffman equivalent code, \(C\), that contains only one synchronising codeword, 01\(^{i-2}0\), of length \(r = m + 1\), if and only if \(C_i = n_i \leq C_i + 0 + 1\) for \(m \leq i \leq M\).

Proof: Obviously, by Corollary B.1, the necessary condition part of the theorem holds.

Notice that although Lemmas B.1–B.4 were derived based on the assumption of always terminating 0-nodes whenever possible, by Theorem 9 of Escott and Perkins (1998), the sufficient condition part of Theorem B.1 still holds. Suppose it does not hold, i.e., the conditions \(C_i \leq n_i \leq C_i + 0 + 1\), for \(m \leq i \leq M\) do not hold, and there still exists a binary Huffman equivalent code, \(C\), that contains only one synchronising codeword 01\(^{i-2}0\) of length \(r = m + 1\). By Theorem 9 of Escott and Perkins (1998), there exists one equivalent code, \(C'\) (which can be obtained by always terminating 0-nodes whenever possible). Then, for such a code, \(C'\), Lemmas B.1–B.4 will hold and the conditions \(C_i \leq n_i \leq C_i + 0 + 1\), for \(m \leq i \leq M\) must hold and we have a contradiction.