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). Chabbouh and Lamy (2002) proposed another VLC tree structure with good synchronisation behaviour. Higgs 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-posterior 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 $A^+ = \cup_{n\geq 1} A^n$ be the set of finite sequences of elements of A and $A^* = A^+ \cup \{\lambda\}$, where λ 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 $(n_1, \ldots, n_M)$ be the length vector of code C, where n_i 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 $(n_1, \ldots, n_M)$ satisfies

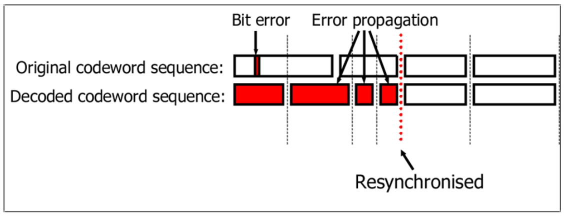
$$\sum_{i=1}^{M} n_i 2^{-i} = 1.$$

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 Error propagation and resynchronisation (see online version for colours)



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 $c_1c_2\ldots c_r = b_{n-r+1}b_{n-r}\ldots b_n$, but $c_1c_2\ldots c_r \neq b_ib_{i+1}\ldots b_{i+r-1}$ 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 $z^*c_1c_2\ldots c_r$, where $z^* \in A^*$. A node, N, is a d-node if its path is of the type $z^*c_1c_2\ldots c_k$ 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* $c_1c_2 \ldots c_r$ *be a synchronising codeword of C such that* $r = m + 1$ *and* $c_1 = 0$ (*resp.* 1). *Then,* $c_i = 1$ (*resp.* 0) *for* $i = 2, \ldots, r - 1$. *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*.

CF_i: The number of level *i c*-nodes in the *FBT*.

DF_i: The number of level *i d*-nodes in the *FBT*.

$C0_i$: The number of level *i c*-nodes in an *SFBT* whose root is a 0-node.

Cc_i: 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.

$D0_i$: The number of level *i d*-nodes in an *SFBT* whose root is a 0-node.

Dc_i: 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.

$T0_i$: 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 $(n_1, \ldots, n_M)$ 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*

$$\begin{cases} C_i = CF_i \quad \text{for } 1 \leq i < 2m \\ C_i = CF_i - \sum_{k=m}^{i-m} \left(T0_k \times C0_{i-k}\right) - \sum_{k=m}^{i-m} \left(\dfrac{C_k}{2} \times Cc_{i-k}\right) \\ \qquad \text{for } i \geq 2m, \end{cases}$$

where

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

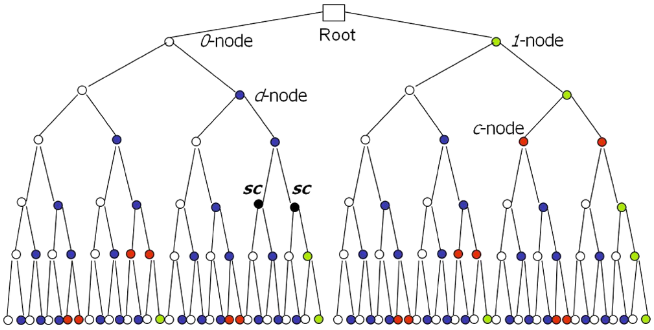
$$\begin{cases} C0_i = 0 & \text{for } 1 \leq i < m \\ C0_i = 2 & \text{for } i = m \\ C0_i = 2^{i-(m+1)} \times 2 = 2^{i-m} & \text{for } i > m, \end{cases} \tag{1-2}$$

and

$$\begin{cases} Cc_i = 0 & \text{for } 1 \leq i < m \\ Cc_i = 2 & \text{for } i = m \\ Cc_i = 2^{i-(m+1)} \times 2 \times 2 = 2^{i-m+1} & \text{for } i > m. \end{cases} \tag{1-3}$$

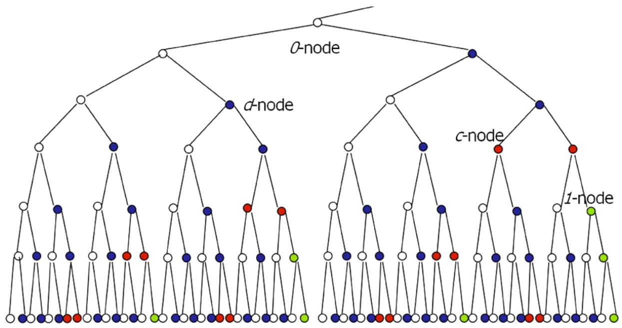
Proof: (1-1) The number of level *i* ($i < m$) *c*-nodes in the *FBT* is trivially equal to zero. The paths of the two level *m c*-nodes in the *FBT* are $1^{r-2}0$ and 1^{r-1}, respectively. The paths of the level *i* ($i > m$) *c*-nodes in the *FBT* are of the form $x_1x_2\ldots x_{i-r}01^{r-2}0$ and $x_1x_2\ldots x_{i-r}01^{r-1}$, 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 $2^{i-(m+1)} \times 2 = 2^{i-m}$ 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 $1^{r-2}0$ and 1^{r-1}, respectively. The paths of those level i $(i > m)$ c-nodes in the *SFBT* are also of the form $x_1x_2\ldots x_{i-r}01^{r-2}0$ and $x_1x_2\ldots x_{i-r}01^{r-1}$ (see Figure 3).

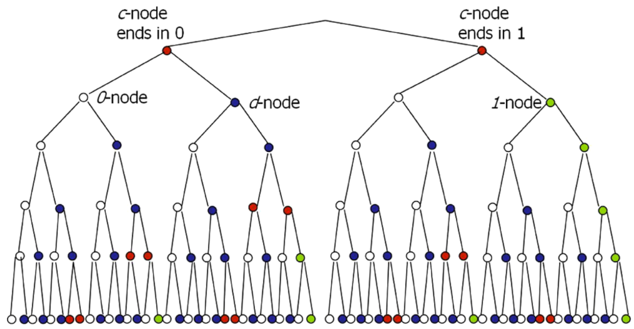
Figure 3 Any one *SFBT* of depth 6 whose root is 0-node when two shortest synchronising codewords, 0110 and 0111, exist in code C (see online version for colours)



(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 $1^{r-2}0$ and 1^{r-1}. 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 $x_1x_2\ldots x_{i-r}01^{r-2}0$ and $x_1x_2\ldots x_{i-r}01^{r-1}$. Hence, the total number of level i c-nodes in the pair of *SFBTs* is equal to $2^{i-(m+1)} \times 2 \times 2 = 2^{i-m+1}$ 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 $1 \le i \le M$. □

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*

$$\begin{cases} D_i = DF_i & \text{for } 1 \le i \le m \text{ and} \\ D_i = DF_i - \sum_{k=m}^{i-1}\left(T0_k \times D0_{i-k}\right) - \sum_{k=m}^{i-1}\left(\frac{C_k}{2} \times Dc_{i-k}\right) & \\ & \text{for } i > m, \end{cases}$$

where

$$\begin{cases} DF_i = 0 & \text{for } i = 1 \\ DF_i = \sum_{k=2}^{i} 2^{i-k} & \text{for } 1 < i < m \\ DF_i = \sum_{k=2}^{m} 2^{i-k} & \text{for } i \ge m, \end{cases} \tag{2-1}$$

$$\begin{cases} D0_i = 1 & \text{for } i = 1 \\ D0_i = \sum_{k=2}^{i} 2^{i-k} + 1 & \text{for } 1 < i < m \\ D0_i = \sum_{k=2}^{m} 2^{i-k} & \text{for } i \ge m, \end{cases} \tag{2-2}$$

and

$$\begin{cases} Dc_i = 1 & \text{for } i = 1 \\ Dc_i = \sum_{k=2}^{i} 2^{i-k} \times 2 + 1 = \sum_{k=1}^{i} 2^{i-k} & \text{for } 1 < i < m \\ Dc_i = \sum_{k=2}^{m} 2^{i-k} \times 2 = \sum_{k=1}^{m-1} 2^{i-k} & \text{for } i \ge m. \end{cases} \tag{2-3}$$

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 $x_1x_2\ldots x_{i-j-1}01^j$, where $j = 1\ldots i-1$. Hence, the number of level i d-nodes in the *FBT* is equal to

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

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

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

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

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

(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^i and $x_1x_2\ldots x_{i-j-1}01^j$, 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 $x_1x_2\ldots x_{i-j-1}01^j$, $j = 1\ldots i-1$. Thus, the total number of level i d-nodes in the pair of *SFBTs* is equal to

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

The paths of the level i ($i \geq m$) d-nodes in either of these two *SFBTs* are all of the form $x_1x_2\ldots x_{i-j-1}01^j$, where $j = 1\ldots m-1$. Hence, the total number of level i d-nodes in the pair of *SFBTs* is equal to

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

From equations (2-1)–(2-3) and $T0_k$, 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_i) *and the number of level i* 0*-nodes taken as codewords* ($T0_i$) *in the BT can be obtained as*

$$\begin{cases} 0_i = 2^i - D_i - 1 & \text{for } 1 \leq i < m \\ 0_i = 2^i - C_i - D_i & \text{for } i = m \\ 0_i = \left(D_{i-1} - \dfrac{C_i}{2}\right) + \left(0_{i-1} - T0_{i-1}\right) & \text{for } m < i \leq M \end{cases} \qquad (3\text{-}1)$$

and

$$\begin{cases} T0_i = 0 & \text{for } 1 \leq i < m \\ T0_i = n_i - C_i & \text{for } m \leq i \leq M \end{cases}. \qquad (3\text{-}2)$$

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}–$T0_{i-1}$ (which is the number of extended 0-nodes at level $i-1$) and $D_{i-1} - C_i/2$ (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* ($n_1, \ldots, n_M$), *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,* $01^{r-2}0$ *and* 01^{r-1}, *with* $r = m+1$, *if and only if* $C_i \leq n_i \leq C_i + 0_i$ *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 $C_i + 0_i$; otherwise, some d-nodes will be terminated. Therefore, $C_i \leq n_i \leq C_i + 0_i$ 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, $C_i \leq n_i \leq C_i + 0_i$ 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*,

$$\begin{cases} (E_1): C_i < n_i < C_i + 0_i + 1_i,\ 0_i > 0 \text{ and } 1_i > 0; \\ (E_2): n_{i+1} < C_{i+1} + 0_{i+1} + 1_{i+1}; \\ \quad\vdots \\ (E_m): n_{i+m-1} < C_{i+m-1} + 0_{i+m-1} + 1_{i+m-1}; \\ (E_{m+1}): C_{i+m} < n_{i+m}; \end{cases} \tag{4}$$

hold for some i, *where* $m \le i \le M\text{-}m$, *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*, $01^{r-2}0$, *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 ($T0_i > 0$, i.e., $n_i > C_i$ & $0_i > 0$) and at least one 1-node can be extended ($T1_i < 1_i$, i.e., $T1_i = n_i - C_i - 0_i < 1_i$).

(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-$(i + r - 1)$ 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 ($T1_{i+h} < 1_i$, i.e., $T1_{i+h} = n_{i+h} - C_{i+h} - 0_{i+h} < 1_{i+h}$) exists for each of the following levels, $(i + h)$ for $1 \le h \le m - 1$, 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).

$$\begin{cases} 1_{i+1} = 1_{i+1} - 1; \\ 1_{i+2} = 1_{i+2} - 1; \\ \quad\vdots \\ 1_{i+m-1} = 1_{i+m-1} - 1; \\ 0_{i+m} = 0_{i+m} - 1 \text{ and } C_{i+m} = C_{i+m} + 1; \end{cases} \tag{5}$$

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 $1 \le i < m$ and $n_m \ge 1$ for some $m > 1$.

Output: A synchronous binary Huffman equivalent code, C.

Step 1: Let m be the smallest integer satisfying $n_m \ne 0$, and let M be the largest integer satisfying $n_M \ne 0$.
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 $1 \le i \le M$ {
Terminate all level i c-nodes.
Extend all level i d-nodes.
If $i \ge m$, 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 $1 \le i \le M$ {
Terminate all level i c-nodes.
Extend all level i d-nodes.
If $m \le i \le M\text{-}m$ then {
Swapno = 0
Repeat {
If equation (4) holds & (swapno < 1_i), then {terminate any one 1-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
}
else if $M–m < i \le M$ 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

Level i	n_i	C_i	0_i
3	2	2	3
4	7	2	5
5	7	4	3
6	5	6	0

Table 2 Test of Theorem A.1

Level i	n_i	C_i	0_i
3	2	1	4
4	7	1	6
5	7	2	5
6	5	3	3
7	1	0	1
8	1	0	1
9	1	1	1
10	2	0	1

Table 3 Test of Theorem B.1

Level i	n_i	C_i	0_i	1_i
3	2	1	3	1
4	7	1	5	2
5	7	2	3	3
6	5	2	1	3
7	1	0	1	1
8	1	0	1	1
9	1	0	1	1
10	2	0	1	1

Code C is shown in Table 4. We also list the codes, respectively, obtained by using the algorithms of Ferguson and Rabinowiz (1984), Escott and Perkins (1998) and Zhou and Zhang (2002). Although the codes generated from Algorithm 1 and the Fixed Order method both have the same number of synchronising codewords, the latter has a smaller MEPL and VEPL. In general, a larger sum for the probabilities of transmitting a synchronising codeword leads to a quicker resynchronisation for the code. The sum of the probabilities of transmitting a synchronising codeword for the above-mentioned two codes are, respectively, equal to 0.2593 and 0.2919. Although the sum of the probabilities of transmitting a synchronising codeword for the code constructed by the Max Gain method is smaller (0.2563), the code also has a smaller MEPL and VEPL. Thus, it is a better statistically synchronisable code. That is, using the MEPL and VEPL to evaluate the synchronisation capability of a code is more accurate than using the sum of the probabilities of transmitting a synchronising codeword.

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

Letter	*Prob.*	*Fixed Order of Zhou and Zhang (2002)*	*Max Gain of Zhou and Zhang (2002)*	*Ferguson and Rabinowiz (1984)*	*Escott and Perkins (1998)*	*Algorithm 1*
E	0.1278	001	001	001	110	110
T	0.0855	101	101	010	100	111
O	0.0804	0001	1001*	0001	0110*	0110*
A	0.0778	1001*	1101	0110	0000	0000
N	0.0686	0101	0001	0111	0010	0010
I	0.0667	1101	0101	1010	0100	0100
R	0.0651	0111	0111	1011	0111	0111
S	0.0622	1111	1111	1100	1010	1000
H	0.0595	0000	0000	1111	1110	1010
D	0.0404	10001*	01001*	00001	00110*	00110*

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

Letter	*Prob.*	*Fixed Order of Zhou and Zhang (2002)*	*Max Gain of Zhou and Zhang (2002)*	*Ferguson and Rabinowiz (1984)*	*Escott and Perkins (1998)*	*Algorithm 1*
L	0.0372	01001*	11001*	10010	10110*	10110*
U	0.0308	11001*	11101	10011	00010	00010
C	0.0296	01101*	01101*	11010	01010	00111
M	0.0288	11101	10001	11011	10111	01010
P	0.0223	10000	10000	11100	11110	10010
F	0.0197	01000	11000	11101	11111	10111*
Y	0.0196	110001*	111001*	100001*	000110*	000110*
W	0.0176	011001*	011001*	100010	010110*	010110*
G	0.0174	111001*	010001*	100011	000111	100110*
B	0.0141	110000*	010000*	000001	001110	000111*
V	0.0112	011000	011000	000000	010111	010111*
K	0.0074	1110001*	1110001	1000001*	0011110	1001111
J	0.0051	11100001	11100001	10000001	00111110	10011100
X	0.0027	111000001	111000001	100000001	001111110	100111010
Z	0.0017	1110000001	1110000001	1000000001	0011111110*	1001110110*
Q	0.0008	1110000000	1110000000	1000000000	0011111111	1001110111
MEPL		1.9030	1.8483	3.3998	2.0522	1.9260
VEPL		1.3634	1.2294	7.2986	2.0672	1.5225

*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.

From Examples 1 and 2, we find that none of the existing algorithms can be guaranteed to find an optimal solution. That is, all of the existing algorithms are just heuristic, and an algorithm for finding the Huffman equivalent code with minimum MEPL and VEPL is still not available, and may be impossible, as stated by Zhou and Zhang (2002).

A shorter synchronising codeword (i.e., its occurrence probability is higher) will result in a quicker resynchronisation for the code. We conjecture that, for a given length vector, if only one shortest synchronising codeword of length *r* can exist in the corresponding constructed code, then the two codes, respectively, generated by the Fixed Order and Max Gain methods of Zhou and Zhang (2002) will have better synchronisation capability than that generated by Algorithm 1. However, if two synchronising codewords with the shortest length, r, can exist simultaneously in the corresponding constructed code, the code generated by Algorithm 1 will have better synchronisation capability than the two codes, respectively, generated by the Fixed Order and Max Gain methods of Zhou and Zhang (2002).

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

Letter	*Prob.*	*F.O. of Zhou and Zhang (2002)*	*M.G. of Zhou and Zhang (2002)*	*Algorithm 1*
A	0.25	01	01	10
B	0.25	11	11	11
C	0.125	101*	101*	010*
D	0.125	001	001	011*
E	0.125	000	000	000
F	0.0625	1001*	1001*	0010*
G	0.0625	1000*	1000*	0011*
MEPL		1.9252	1.9252	1.8163
VEPL		1.1318	1.1318	0.9637

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

Cao, L., Yao, L. and Chen, C.W. (2007) 'MAP decoding of variable length codes with self-synchronization strings', *IEEE Trans. Signal Processing*, Vol. 55, No. 8, August, pp.4325–4330.

Capocelli, R.M., Gargano, L., and Vaccaro, U. (1988) 'On the characterization of statistically synchronizable variable-length codes', *IEEE Trans. Inf. Theory*, Vol. 34, July, pp.817–825.

Capocelli, R.M., Santis, A.A.D., Gargano, L. and Vaccaro, U. (1992) 'On the construction of statistically synchronizable codes', *IEEE Trans. Inf. Theory*, Vol. 38, March, pp.407–414.

Chabbouh, S. and Lamy, C. (2002) 'A structure for fast synchronizing variable-length codes', *IEEE Commun. Letters*, Vol. 6, No. 11, November, pp.500–502.

Escott, A.E. and Perkins, S. (1996) 'The construction of binary Huffman equivalent codes with two short synchronizing codewords', *Proc. Int. Symp. Inf. Theory and its Applications*, Victoria, Canada, pp.294–297.

Escott, A.E. and Perkins, S. (1998) 'Binary Huffman equivalent codes with a short synchronizing codeword', *IEEE Trans. Inf. Theory*, Vol. 44, January, pp.346–351.

Ferguson, T.J. and Rabinowiz, J.H. (1984) 'Self-synchronizing Huffman codes', *IEEE Trans. Inf. Theory*, Vol. IT-30, July, pp.687–693.

Higgs, M.B.J., Perkins, S. and Smith, D.H. (2009) 'The construction of variable length codes with good synchronization properties', *IEEE Trans. Inf. Theory*, Vol. 55, No. 4, April, pp.1696–1700.

Huang, Y-M. and Wu, S-C. (2003) 'Shortest synchronizing codeword of a binary Huffman equivalent code', *Proc. IEEE Int. Conf. on Information Technology: Coding and Computing (ITCC)*, Las Vegas, IEEE Computer Society, April, pp.226–231.

Huffman, D.A. (1952) 'A method for the construction of minimum redundancy codes', *Proc. IRE*, Vol. 40, No. 2, pp.1098–1101.

Kleinrock, L. (1975) *Queueing System Volume I: Theory*, John Wiley & Sons Incorporation, New York.

Malinowski, S., Jegou, H. and Guillemot, C. (2007) 'Synchronization recovery and state model reduction for soft decoding of variable length codes', *IEEE Trans. Inf. Theory*, Vol. 53, No. 1, January, pp.368–377.

Montgomery, B.L. and Abrahams, J. (1986) 'Synchronization of binary source codes', *IEEE Trans. Inf. Theory*, Vol. IT-32, November, pp.849–854.

Perkins, S. and Escott, A. E. (1999) 'Synchronizing codewords for *q*-ary Huffman codes', *Discrete Mathematics*, Vols. 197, 198, February, pp.637–655.

Rudner, B. (1971) 'Construction of minimum-redundancy codes with an optimum synchronization property', *IEEE Trans. Inf. Theory*, Vol. IT-17, July, pp.478–487.

Takishima, Y., Wada, M. and Murakami, H. (1994) 'Error states and synchronization recovery for variable length codes', *IEEE Trans. Commun.*, Vol. 42, Nos. 2–4, February–March, pp.782–792.

Wei, V.K.W. and Sholtz, R.A. (1980) 'On the characterization of statistically synchronizable codes', *IEEE Trans. Inf. Theory*, Vol. IT-26, November, pp.733–735.

Zhou, G. and Zhang, Z. (2002) 'Synchronization recovery of variable-length codes', *IEEE Trans. Inf. Theory*, Vol. 48, No. 1, January, pp.219–227.

Zhou, J., Au, O.C., Fan, X. and Wong, P.H.W. (2008) 'Error recovery of variable length codes over BSC with arbitrary crossover probability', *Proc. IEEE Int. Symp. Inf. Theory*, Toronto, Canada, July, pp.1188–1192.

Appendices

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

Cc_i: The number of level i *c*-nodes in an *SFBT* whose root is a *c*-node.

Dc_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*

$$\begin{cases} C_i = CF_i & \text{for } 1 \leq i < 2m \\ C_i = CF_i - \sum_{k=m}^{i-m} (T0_k \times C0_{i-k}) - \sum_{k=m}^{i-m} (C_k \times Cc_{i-k}) & \\ & \text{for } i \geq 2m, \end{cases}$$

where

$$\begin{cases} CF_i = 0 & \text{for } 1 \leq i < m \\ CF_i = 1 & \text{for } i = m \\ CF_i = 2^{i-(m+1)} & \text{for } i > m, \end{cases} \quad \text{(A1-1)}$$

$$\begin{cases} C0_i = 0 & \text{for } 1 \leq i < m \\ C0_i = 1 & \text{for } i = m \\ C0_i = 2^{i-(m+1)} & \text{for } i > m, \end{cases} \quad \text{(A1-2)}$$

and

$$\begin{cases} Cc_i = 0 & \text{for } 1 \le i \le m \\ Cc_i = 2^{i-(m+1)} & \text{for } i > m. \end{cases} \quad \text{(A1-3)}$$

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 $1 \le i \le m$. The path of the level i ($i > m$) c-nodes in the *SFBT* is of the form $x_1x_2,\ldots x_{i-r}01^{r-1}$. Hence, the number of level i c-nodes in the *SFBT* is equal to $2^{i-(m+1)}$ for $i > m$. □

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

$$\begin{cases} D_i = DF_i & \text{for } 1 \le i \le m \\ D_i = DF_i - \sum_{k=m}^{i-1}(T0_k \times D0_{i-k}) - \sum_{k=m}^{i-1}(C_k \times Dc_{i-k}) \\ \quad \text{for } i > m, \end{cases}$$

where

$$\begin{cases} DF_i = 0 & \text{for } i = 1 \\ DF_i = \sum_{k=2}^{i} 2^{i-k} & \text{for } 1 < i < m \\ DF_i = \sum_{k=2}^{m} 2^{i-k} & \text{for } i \ge m. \end{cases} \quad \text{(A2-1)}$$

$$\begin{cases} D0_i = 1 & \text{for } i = 1 \\ D0_i = \sum_{k=2}^{i} 2^{i-k} + 1 & \text{for } 1 < i < m \\ D0_i = \sum_{k=2}^{m} 2^{i-k} & \text{for } i \ge m, \end{cases} \quad \text{(A2-2)}$$

and

$$\begin{cases} Dc_i = 0 & \text{for } i = 1 \\ Dc_i = \sum_{k=2}^{i} 2^{i-k} & \text{for } 1 < i < m \\ Dc_i = \sum_{k=2}^{m} 2^{i-k} & \text{for } i \ge m. \end{cases} \quad \text{(A2-3)}$$

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 $x_1x_2 \ldots x_{i-j-1}01^j$ (x_1, x_2, …, $x_{i-j-1} \in \{0,1\}$), where $j = 1 \ldots i-1$. Hence, the number of level i d-nodes in the *SFBT* is equal to

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

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

$$\sum_{k=2}^{m} 2^{i-k} \quad \text{for } i \ge m.$$ □

Lemma A.3: *The number of level i* 0*-nodes* (0_i) *and the number of level i* 0*-nodes taken as codewords* ($T0_i$) *in the BT can be obtained as*

$$\begin{cases} 0_i = 2^i - D_i - 1 & \text{for } 1 \le i < m \\ 0_i = 2^i - C_i - D_i & \text{for } i = m \\ 0_i = D_{i-1} + (0_{i-1} - T0_{i-1}) & \text{for } m < i \le M \end{cases} \quad \text{(A3-1)}$$

and

$$\begin{cases} T0_i = 0 & \text{for } 1 \le i < m \\ T0_i = n_i - C_i & \text{for } m \le i \le M. \end{cases} \quad \text{(A3-2)}$$

Proof: (A.3-1) On the basis of Escott and Perkins (1998), extending a level i 0-node or extending a level i d-node must form a level-(i + 1) 0-node for $m \le i \le M$–1. Hence, for $m < i \le M$, 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 $i-1$). □

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* $n_i = 0$ *for* $i < m$ *and* $n_m \ge 1$ *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* $C_i \le n_i \le C_i + 0_i$ *for* $m \le i \le$ 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 $C_i + 0_i$; otherwise, some d-nodes will be terminated. Therefore, $C_i \le n_i \le C_i + 0_i$ for $m \le i \le M$ and the sufficient condition part of the theorem is proved.

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

(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_i: The number of level i 1-nodes in the *BT*.

$T1_i$: The number of level i 1-nodes taken as codewords in the *FBT*.

Let $(n_1, \ldots, n_M)$ be the length vector of any binary Huffman equivalent code, C, with only one synchronising codeword $01^{r-2}0$, 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*

$$\begin{cases} C_i = CF_i & \text{for } 1 \leq i < 2m \\ C_i = CF_i - \sum_{k=m}^{i-m}(T0_k \times C0_{i-k}) \\ \quad - \sum_{k=m}^{i-m}(C_k \times Cc_{i-k}) - \sum_{k=m}^{i-m}(T1_k \times C1_{i-k}) \\ \quad \text{for } i \geq 2m, \end{cases}$$

where

$$\begin{cases} CF_i = 0 & \text{for } 1 \leq i < m \\ CF_i = 1 & \text{for } i = m \\ CF_i = 2^{i-(m+1)} & \text{for } i > m, \end{cases} \tag{B1-1}$$

$$\begin{cases} C0_i = 0 & \text{for } 1 \leq i < m \\ C0_i = 1 & \text{for } i = m \\ C0_i = 2^{i-(m+1)} & \text{for } i > m, \end{cases} \tag{B1-2}$$

$$\begin{cases} C1_i = 0 & \text{for } 1 \leq i \leq m \\ C1_i = 2^{i-(m+1)} & \text{for } i > m, \end{cases} \tag{B1-3}$$

and

$$\begin{cases} Cc_i = 0 & \text{for } 1 \leq i < m \\ Cc_i = 1 & \text{for } i = m \\ Cc_i = 2^{i-(m+1)} & \text{for } i > m. \end{cases} \tag{B1-4}$$

Proof: (B1-4) The number of level i c-nodes in an *SFBT* whose root is a c-node is trivially equal to zero for $1 \leq i < m$. The path of the level i ($i > m$) c-nodes in the *SFBT* is of the form $x_1x_2\ldots x_{i-r}01^{r-1}$. Hence, the number of level i c-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*

$$\begin{cases} D_i = DF_i & \text{for } 1 \leq i \leq m \\ D_i = DF_i - \sum_{k=m}^{i-1}(T0_k \times D0_{i-k}) - \sum_{k=m}^{i-1}(C_k \times Dc_{i-k}) \\ \quad - \sum_{k=m}^{i-1}(T1_k \times D1_{i-k}) & \text{for } i > m, \end{cases}$$

where

$$\begin{cases} DF_i = 0 & \text{for } i = 1 \\ DF_i = \sum_{k=2}^{i} 2^{i-k} & \text{for } 1 < i < m \\ DF_i = \sum_{k=2}^{m} 2^{i-k} & \text{for } i \geq m, \end{cases} \tag{B2-1}$$

$$\begin{cases} D0_i = 1 & \text{for } i = 1 \\ D0_i = \sum_{k=2}^{i} 2^{i-k} + 1 & \text{for } 1 < i < m \\ D0_i = \sum_{k=2}^{m} 2^{i-k} & \text{for } i \geq m, \end{cases} \tag{B2-2}$$

$$\begin{cases} D1_i = 0 & \text{for } i = 1 \\ D1_i = \sum_{k=2}^{i} 2^{i-k} & \text{for } 1 < i < m \\ D1_i = \sum_{k=2}^{m} 2^{i-k} & \text{for } i \geq m, \end{cases} \tag{B2-3}$$

and

$$\begin{cases} Dc_i = 1 & \text{for } i = 1 \\ Dc_i = \sum_{k=2}^{i} 2^{i-k} + 1 & \text{for } 1 < i < m \\ Dc_i = \sum_{k=2}^{m} 2^{i-k} & \text{for } i \geq m. \end{cases} \tag{B2-4}$$

Proof: (B2-4) The number of level 1 d-nodes in an *SFBT* whose root is a c-node is trivially equal to one. The paths of the level i ($1 < i < m$) d-nodes in the *SFBT* are of the form 1^i and $x_1x_2\ldots x_{i-j-1}01^j$ ($x_1x_2\ldots x_{i-j-1} \in \{0,1\}$), 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 ($i \geq m$) d-nodes in the *SFBT* are of the form $x_1x_2\ldots x_{i-j-1}01^j$ ($x_1x_2\ldots x_{i-j-1} \in \{0,1\}$), 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* $(T1_i)$ *in the BT can be obtained as*

$$\begin{cases} 1_i = 1 & \text{for } 1 \leq i \leq m \\ 1_i = C_i + (1_{i-1} - T1_{i-1}) & \text{for } m < i \leq M \end{cases} \tag{B3-1}$$

and

$$\begin{cases} T1_i = 0 \quad \text{for } 1 \le i < m \\ T1_i = \begin{cases} 0 & \text{if } C_i \le n_i \le C_i + 0_i \\ n_i - C_i - 0_i & \text{if } C_i + 0_i < n_i \le C_i + 0_i + 1_i \end{cases} \\ \quad \text{for } m \le i \le M. \end{cases} \quad \text{(B3-2)}$$

Proof: (B3-1) There exists only one 1-node whose path is 1^i at each level i ($1 \le i \le m$) in the *BT*. On the basis of Escott and Perkins (1998), extending a level i 1-node or extending a level i d-node with suffix 01^{r-2} forms a level-($i + 1$) 1-node for $m \le i \le M - 1$. Hence, the number of level i 1-nodes is equal to the sum of $1_{i-1} - T1_{i-1}$ (which is the number of extended 1-nodes at level $i - 1$) and C_i (which is the number of level-($i - 1$) d-nodes with suffix 01^{r-2}) for $m < i \le M$.

(B3-2) There exists no terminal node at each level i ($1 \le i < m$) in the *BT*. Hence, the number of level i 1-nodes taken as codewords in the *BT* is equal to zero for $1 \le i < m$. 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 $m \le i \le M$, the number of level i 1-nodes taken as codewords in the *BT* is equal to zero if $C_i \le n_i \le C_i + 0_i$, and $n_i - C_i - 0_i$ if $C_i + 0_i < n_i \le C_i + 0_i + 1_i$. □

Lemma B.4: *The number of level i* 0-*nodes* (0_i) *and the number of level i* 0-*nodes taken as codewords* ($T0_i$) *in the* BT *can be obtained as*

$$\begin{cases} 0_i = 2^i - D_i - 1 & \text{for } 1 \le i < m \\ 0_i = 2^i - C_i - D_i - 1 & \text{for } i = m \\ 0_i = (D_{i-1} - C_i) + (0_{i-1} - T0_{i-1}) & \\ \quad + (1_{i-1} - T1_{i-1}) & \text{for } m < i \le M \end{cases} \quad \text{(B4-1)}$$

and

$$\begin{cases} T0_i = 0 \quad \text{for } 1 \le i < m \\ T0_i = \begin{cases} n_i - C_i & \text{if } C_i \le n_i \le C_i + 0_i \\ 0_i & \text{if } n_i > C_i + 0_i \end{cases} \\ \quad \text{for } m \le i \le M. \end{cases} \quad \text{(B4-2)}$$

Proof: (B4-1) For the *BT*, there exists no c-node at level i ($1 \le i < m$) by Lemma B.1 and there exists only one 1-node at level i ($1 \le i < m$) by Lemma B.3. Hence, the number of level i 0-nodes is equal to $2^i - D_i - 1$ for $1 \le i < m$. Also by Lemma B.3, there exists only one 1-node at level m. Hence, the number of level m c-nodes is equal to $2^i - C_i - D_i - 1$. On the basis of Escott and Perkins (1998), extending a level i d-node with no suffix 01^{r-2}, extending a level i 0-node, or extending a level i 1-node forms a level-($i + 1$) 0-node for $m \le i \le M - 1$. Hence, for $m < i \le M$, the number of level i 0-nodes is equal to the sum of $D_{i-1} - C_i$ (which is the number of level-($i - 1$) d-nodes with no suffix 01^{r-2}), $0_{i-1} - T0_{i-1}$ (which is the number of extended 0-nodes at level $i - 1$), and $1_{i-1} - T1_{i-1}$ (which is the number of extended 1-nodes at level $i - 1$).

(B4-2) In the *BT*, there exists no terminal node at level i ($1 \le i < m$). Hence, the number of level i 0-nodes taken as codewords is equal to zero for $1 \le i < m$. In addition, all of the c-nodes must be taken as codewords. In the constructing algorithm of Escott and Perkins (1998), 0-nodes are always taken as codewords first whenever possible. Hence, for $m \le i \le M$, the number of level i 0-nodes taken as codewords is equal to $n_i - C_i$ if $C_i \le n_i \le C_i + 0_i$, and 0_i if $n_i > C_i + 0_i$. □

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* ($n_1,\ldots,n_M$), *where* $n_i = 0$ *for* $i < m$ *and* $n_m \ge 1$ *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*, $01^{r-2}0$, *of length* $r = m + 1$, *if and only if* $C_i \le n_i \le C_i + 0_i + 1_i$ *for* $m \le i \le M$.

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 $C_i + 0_i + 1_i$; otherwise, some d-nodes will be terminated. Therefore, $C_i \le n_i \le C_i + 0_i + 1_i$ for $m \le i \le M$ and the sufficient condition part of the corollary is proved.

Since Lemmas B.1–B.4 hold for such a code, *C*, the inequalities, $C_i \le n_i \le C_i + 0_i + 1_i$ for $m \le i \le M$, 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* ($n_1,\ldots,n_M$), *where* $n_i = 0$ *for* $i < m$ *and* $n_m \ge 1$ *for some* $m > 1$, *there exists a binary Huffman equivalent code*, *C*, *that contains only one synchronising codeword*, $01^{r-2}0$, *of length* $r = m + 1$, *if and only if* $C_i \le n_i \le C_i + 0_i + 1_i$ *for* $m \le i \le M$.

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 $C_i \le n_i \le C_i + 0_i + 1_i$ for $m \le i \le M$ do not hold, and there still exists a binary Huffman equivalent code, *C*, that contains only one synchronising codeword $01^{r-2}0$ 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 $C_i \le n_i \le C_i + 0_i + 1_i$ for $m \le i \le M$ must hold and we have a contradiction. □