

### 34. Lossless Source Coding - Huffman and Shannon-Fano Coding

The basic objective of *source coding* is to *remove redundancy* in a source. Source coding therefore achieves data *compression* and *reduces* the *transmission rate*. A reduction in transmission rate can lower the cost of a link and enables more users to share the same link. In general, we can compress data without losing information (*lossless source coding*) or compress data with loss of information (*lossy source coding*). Figures 34.1 and 34.2 show a source coding block diagram and some data compression techniques, respectively.

**Figure 34.1** Source coding block diagram.

**Figure 34.2** Classification of some data compression techniques [1].

The *source efficiency* is defined as  $H(X)/H(X)_{max}$ , where

$$H(X) = - \sum_{i=1}^m p_i \log_2 p_i \quad (34.1)$$

$p_i$  is the probability associated with the  $i$ -th symbol, and  $H(X)$  is at a maximum when the source has equal symbol probabilities.

The **Shannon Noiseless Source Coding theorem** states that the average number of binary symbols per source output can be made to approach the entropy of a source. In another words, the source efficiency can be made to approach unity by means of *source coding*.

For sources with equal symbol probabilities, and/or statistically independent to each other, we can simply encode (map) each symbol into a codeword of *block length*  $n$ .

**Example 34.1**

Symbol 'A' maps into codeword 00

Symbol 'B' maps into codeword 01

Symbol 'C' maps into codeword 10

Symbol 'D' maps into codeword 11 .

The length of each codeword is 2.

In some cases, sources with **unequal** symbol probabilities, and/or statistically **non-independent** symbols, may have relatively poor source efficiencies by the above source encoding method (i.e. source efficiency is much less than unity).

**Example 34.2**

The letter *E* in the English language occurs more frequently than the other letters.

If some symbols are more probable than others, then we can take advantage of this property to make the most frequent symbols be associated with a **codeword** of shortest length. This type of encoding is called **variable-length** encoding (eg. Morse code).

**Huffman Encoding**

Huffman encoding is an efficient lossless source encoding method to remove redundancy. It is a variable-length encoding scheme.

The encoding method of a set of symbols may be determined as follows:

1. Arrange the source symbols in descending order of probability.
2. Create a new source with one less symbol by combining (adding) the two lowest-probability symbols.
3. Repeat steps 1 and 2 until a single-symbol source is achieved.
4. Associate a one and a zero with each pair of probabilities so combined.
5. Encode each original source symbol into the binary sequence generated by the various combinations, with the first combination as the least significant digit in the sequence.

**Example 34.3**

Source symbols *A*, *B*, *C*, *D*, and *E* with probabilities 0.4, 0.2, 0.2, 0.1, and 0.1, respectively.

**Figure 34.3** Huffman encoding flow diagram.

- A* encodes into 1
- B* encodes into 01
- C* encodes into 000
- D* encodes into 0010 (--....--- path)
- E* encodes into 0011

**Figure 34.4** Huffman encoding tree diagram.

Note that the Huffman encoding rule is uniquely decodable; i.e., a stream of encoded symbols can be decoded without the need for synchronisation or framing pulses between the binary sequences representing each symbol. The tree diagram confirms this property, since no encoded sequence is a prefix to another/any other valid sequence. For example, the encoded sequence for symbol *C* is not a prefix of any other valid sequence.

### Calculations

$$\text{Source entropy } H(X) = - \sum_{i=1}^m p_i \log_2 p_i .$$

$$H(X) = -0.4 \log_2 0.4 - 0.2(2) \log_2 0.2 - 0.1(2) \log_2 0.1$$

$$H(X) = 2.12 \text{ bits/symbol.}$$

If  $p_1 = p_2 = \dots = p_m$ ,

$$H(X)_{max} = -0.2(5) \log_2 0.2$$

$$H(X)_{max} = 2.322 \text{ bits/symbol .}$$

Source efficiency **before** encoding =  $H(X) / H(X)_{max} = 0.913$  .

The average number of binary digits used to encode each source symbol **after** encoding is given by

$$L = \sum_{i=1}^m p_i l_i \text{ bits/symbol} \quad (34.2)$$

where  $l_i$  is the number of binary digits required to encode symbol  $i$ . For example 34.3,

$$\begin{aligned} L &= (0.4 \times 1) + (0.2 \times 2) + (0.2 \times 3) + (0.1 \times 4) + (0.1 \times 4) \\ &= 2.2. \end{aligned}$$

The source efficiency  $\eta$  **after** encoding is given by

$$\eta = H(X)/L. \quad (34.3)$$

and  $1-\eta$  is defined as the **source redundancy**. For example 34.3,  $\eta = 2.12/2.2 = 0.964$ .

If  $M$  is the number of binary digits required to represent the source without any source coding, the compression ratio is

$$CR = L/M. \quad (34.4)$$

For example 34.3,  $M=3$  and

$$CR = 2.2/3 = 0.73.$$

The compression ratio  $CR$  indicates the proportion by which the bandwidth required to transmit the source over a binary channel can be reduced.

### Summary

- |    |                   |  |
|----|-------------------|--|
| 1. | $H(X)$            | $= - \sum_{i=1}^m p_i \log_2 p_i$  |
| 2. | $H(X)/H(X)_{max}$ | Source efficiency <b>before</b> encoding   |
| 3. | $l_i$             | Length of the encoded binary pattern for symbol $i$  |
| 4. | $L$               | Average number of binary digits used to encode each source symbol <b>after</b> encoding $= \sum_{i=1}^m p_i l_i$ |
| 5. | $\eta = H(X)/L$   | Source efficiency <b>after</b> encoding  |
| 6. | $M$               | Number of binary digits required to represent the source without encoding  |
| 7. | $CR$              | Compression ratio $= L/M$  |

## Shannon-Fano Encoding

Another efficient variable-length encoding scheme is known as *Shannon-Fano encoding*. The Shannon-Fano encoding procedure is as follows:

1. Arrange the source symbols in descending order of probability.
2. Partition the set into two sets that are as close to equiprobable as possible, and assign 0 to the upper set and 1 to the lower set.
3. Repeat step 2, each time partitioning the set with as nearly equal probabilities as possible until further partitioning is not possible.
4. Encode each original source symbol into the binary sequence generated by the partitioning process.

### *Example 34.4*

Source symbols  $A, B, C, D$ , and  $E$  with probabilities 0.4, 0.2, 0.2, 0.1, and 0.1, respectively.

**Figure 34.5** Shannon-Fano encoding flow diagram.

$A$	encodes into 00
$B$	encodes into 01
$C$	encodes into 10
$D$	encodes into 110
$E$	encodes into 111

### Calculations

$$\text{Source entropy } H(X) = - \sum_{i=1}^m p_i \log_2 p_i .$$

$$H(X) = -0.4\log_2 0.4 - 0.2(2)\log_2 0.2 - 0.1(2)\log_2 0.1$$

$$H(X) = 2.12 \text{ bits/symbol.}$$

If  $p_1 = p_2 = \dots = p_m$ ,

$$H(X)_{max} = -0.2(5) \log_2 0.2$$

$$H(X)_{max} = 2.322 \text{ bits/symbol .}$$

Source efficiency **before** encoding =  $H(X) / H(X)_{max} = 0.913$  .

The average number of binary digits used to encode each source symbol **after** Shannon-Fano encoding is

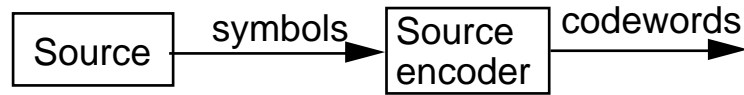
$$\begin{aligned} L &= \sum_{i=1}^m p_i l_i \text{ bits/symbol} \\ &= (0.4 \times 2) + (0.2 \times 2) + (0.2 \times 2) + (0.1 \times 3) + (0.1 \times 3) \\ &= 2.2. \end{aligned}$$

The source efficiency  $\eta$  **after** encoding is

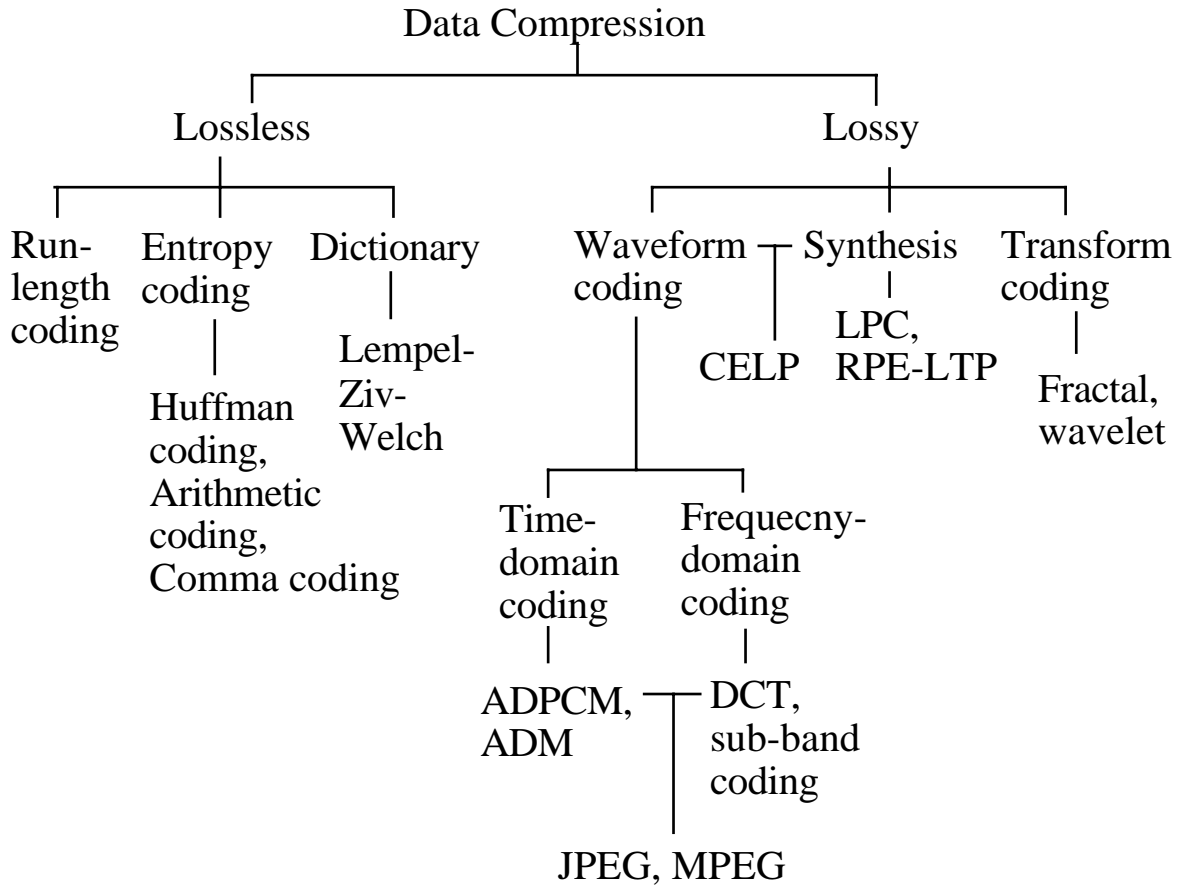
$$\begin{aligned} \eta &= H(X)/L \\ &= 2.12/2.2 = 0.964. \end{aligned}$$

## References

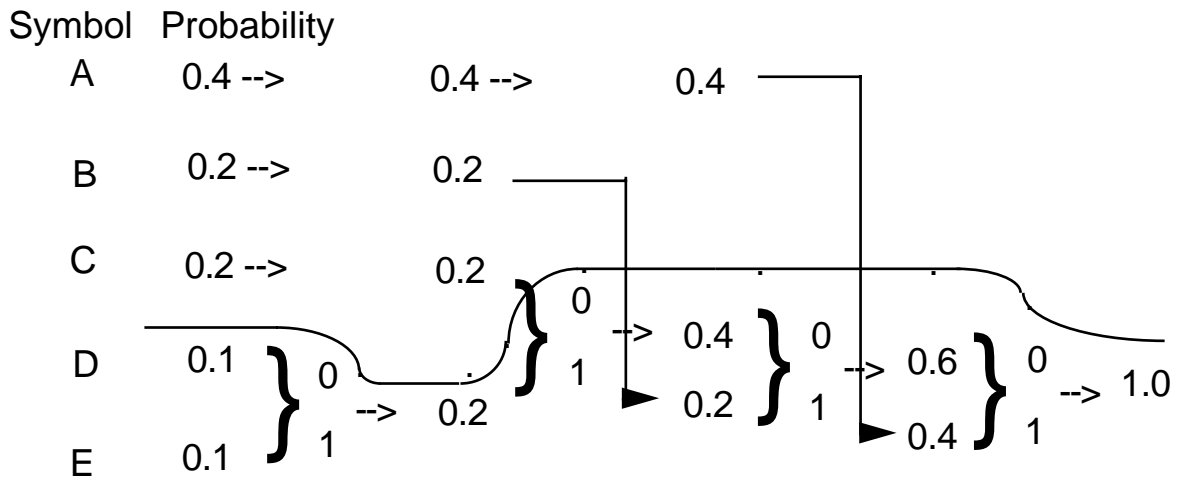
- [1] Wade, G., Coding Techniques: An Introduction to Compression and Error Control, Palgrave, 2000.
- [2] Shanmugam, K. S., 'Digital and Analog Communication Systems', John Wiley & Sons, 1979.
- [3] Hsu, H. P., Analog and Digital Communications, McGraw-Hill, 1993.



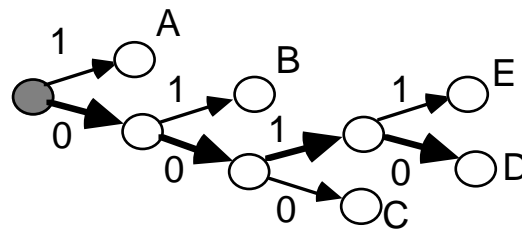
**Figure 34.1** Source coding block diagram.



**Figure 34.2** Classification of some data compression techniques.



**Figure 34.3** Huffman encoding flow diagram.



**Figure 34.4** Huffman encoding tree diagram.

Symbol	Probability	Step 1	Step 2	Step 3	Codeword
A	0.4	0	0		00
B	0.2	0	1		01
C	0.2	1	0		10
D	0.1	1	1	0	110
E	0.1	1	1	1	111

**Figure 34.5** Shannon-Fano encoding.