World Scientific Research Journal (WSRJ) ISSN: 2472-3703
www.wsr-j.org

Application and Analysis of Lossless Compression of Telemetry Data

Tao Yang [a], Jie Dong [b]

School of Shandong University of Science and Technolog University, Qingdao 266590, China

[a]734981537@qq.com, [b]1374623537@qq.com

Abstract: In recent years, the development of space science and technology has been rapid. The development of the aerospace industry has put forward higher requirements for the lossless compression of telemetry data. How to quickly and efficiently perform lossless compression and transmission of data has become a crucial issue in deep space exploration missions. ring. This article describes Huffman coding and LZW coding algorithms in traditional compression methods.
Keywords: Lossless data compression; MATLAB; Huffman coding; LZW coding.

---

1. HUFFMAN CODING

The traditional lossless compression methods are Huffman coding, LZW coding, Shannon-Fano coding, RLE coding, dictionary coding, etc. [1]. This article focuses on Huffman coding and LZW coding.

Huffman coding is Huffman coding, which is a variable-length coding method. In variable-length encoding, the encoding output of the encoder is a codeword of unequal word length. According to the statistical probability of the information symbol in the input data, the output data is allocated in different bytes, such as large-probability information symbols. The short output word length is assigned, the relatively low probability information symbol, and its long output word length. In this way, the information in the data can be coded according to the size of the probability [2] of the probability of occurrence of different codeword lengths. The process of constructing the optimal binary tree using the Huffman tree is as follows:

(1) Considering n weighted leaves as a set of binary trees, and arranging the n weighted leaves in descending order of weights;

(2) Extract two binary trees with the smallest and the second least weight, and add a common predecessor to the root node of the binary tree to form a new binary tree whose root defines a weight value with two words. The sum, put the new binary tree into the collection;

(3) Repeat the first two steps until there is only one binary tree left in the collection, which is the optimal solution.

Let the left leaf of each node represent the character "0", the right leaf represents the character "1", so that the score character on the path from the root node to the leaf node can be formed as

the code of the leaf node. For all the characters that need to be transmitted, let each leaf correspond to a leaf node. The weight value is the frequency of its appearance. According to the Huffman algorithm, it is possible to invert the binary code of each character.

$$U = \begin{pmatrix} a1 & a2 & a3 & a4 & a5 & a6 & a7 \\ 0.20 & 0.19 & 0.18 & 0.17 & 0.15 & 0.10 & 0.01 \end{pmatrix}$$

Using Huffman coding, as shown in Figure 1.1. The codeword corresponding to a single message can be divided by the vertex, that is, the root node divides the path taken by each endpoint ak.

Average code length:

$$\bar{n} = \sum_{k=1}^{k} p_k n_k = p_1 n_1 + p_2 n_2 + p_3 n_3 + p_4 n_4 + p_5 n_5 + p_6 n_6 + p_7 n_7 = 2.7$$

Entropy of the source:

$$H(U) = -\sum_{k=1}^{7} p(a_k) log p(a_k) = 2.61$$

Coding efficiency:

$$\eta = \frac{H(U)}{\bar{n}} = \frac{2.61}{2.72} = 96\%$$

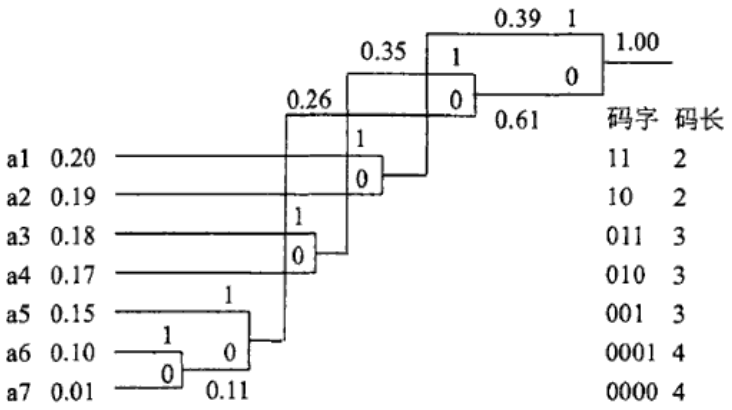

Figure 1. Huffman coding

Through the introduction of the above Huffman coding, it can be seen that each character node is a leaf node. The coding method is: defining a code '0' from the root node to the left, and a '1' to the right, and traversing the leaf node. The resulting binary code string is the coded value of this character. Since the character code word is a prefix code, each character can be uniquely decoded in the decoding process with reference to the Huffman tree, but the disadvantage of the prefix code is that the error has a propagation function, when there is a 1-bit code word error, thereafter The decoding process is probably not correct.

2. LZW CODING

LZW algorithm belongs to dictionary coding, according to the characteristics of the data itself contains a duplicate code, use the code of the string that has appeared in front to replace the

content of the same string behind, in order to achieve data compression. Compression based on this algorithm does not output a single character, only the code in the dictionary string, so the dictionary cannot be empty at the beginning and should contain all single bytes that may appear in the character [3].

LZW encoding can make use of the redundancy of the frequency of occurrence of characters. Only one scan is performed on the characters, and the statistics on the input data are not verified. Three important objects of the LZW compression algorithm are: data stream, coded stream, and compiled table.

Let the source sequence u of length L be divided into M(u) segment code segments. The binary code symbol length of each phrase is [log M(u)+log K], so the total code length after u encoding is M (u)•([log M(u)+logK]), the average code length is:

$$\bar{n}(\mathrm{u}) = \frac{\mathrm{M(u)} \cdot ([\log \mathrm{M(u)}] + \log [\mathrm{K}])}{L}$$

After simplifying:

$$\frac{\mathrm{M(u)} \cdot (\log \mathrm{M(u)} + \log\ \mathrm{K})}{L} \leq \bar{n}(\mathrm{u}) < \frac{\mathrm{M(u)} \cdot (\log \mathrm{M(u)} + \log \mathrm{K} + 2)}{L}$$

Let the longest segment be lmax, then:

$$\mathrm{M(u)} = \sum_{l=1}^{lmax} K^l = \frac{K^{lmax+1} - K}{K - 1}$$

$$\mathrm{L} = \sum_{l=1}^{lmax} lK^l = \frac{K}{(K-1)^2}\{lmax\, K^{lmax+1} - (lmax + 1)K^{lmax+1}\}$$

When lmax is large enough, the approximation is

$$\mathrm{M(u)} \approx \frac{K^{lmax+1}}{K - 1}$$

Therefore:

$$\frac{logM(u)}{lmax} + \frac{logK}{lmax} \leq \bar{n}(\mathrm{u}) < \frac{logM(u)}{lmax} + \frac{logK + 2}{lmax}$$

The distribution probability of the source symbols is P(ak), k=1, 2,...,k. then

$$\mathrm{Mlmax} = \frac{lmax!}{\prod_k (P(a_k)lmax)!} \approx \frac{l_{max}^{lmax}}{\prod_k (P(a_k)lmax)^{p(a_k)lmax}}$$

Logarithm

$L \approx lmax\, M_{lmax} = lmax 2^{lmaxH(U)}$

To remove the shorter segment, then:

$\mathrm{L} \approx lmax\, M_{lmax} = lmax\, 2^{lmaxH(U)}$

Then the average code length tightens rain source entropy.

The LZW compression data algorithm is a dictionary-based compression method. The relationship between compression ratio and dictionary space is shown in Figure 1.3.

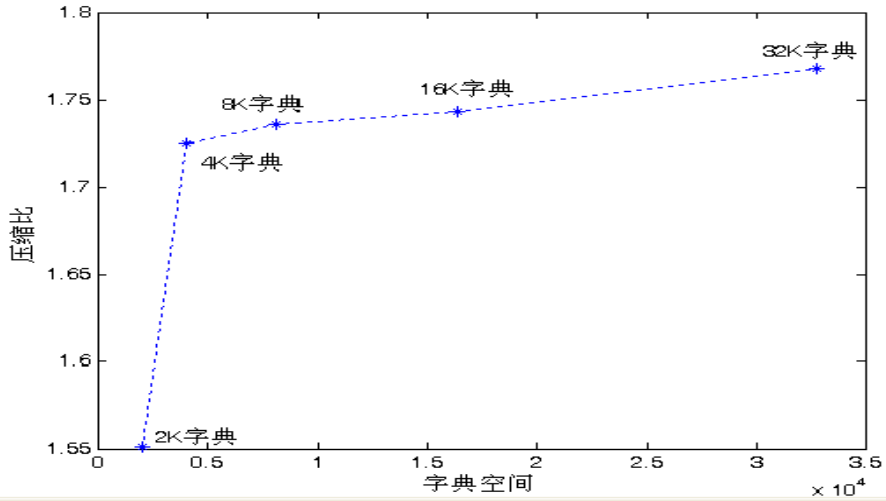

Figure 2. The relationship between compression ratio and dictionary space

The LZW compression algorithm does not need to know in advance the statistical characteristics of the information in the data to be compressed, thus reducing the difficulty of the compression algorithm. Compared to other data compression algorithms, LZW accomplishes data compression by searching the dictionary, so it is easier to implement by hardware. However, in the initial stage of data compression, LZW compression method, because the string table stored in the dictionary is limited, sometimes the compressed file is larger than the original file; and each time the newly received data is compressed, it must be from the dictionary. The first character begins to be compared one by one, which will affect the time of LZW dictionary compression, thereby reducing the efficiency of the dictionary compression.

With the rapid development of aerospace science and technology, the explosive growth of aerospace telemetry data puts forward higher requirements on the lossless compression technology of telemetry data. How to quickly and efficiently perform lossless compression and transmission of data has become a crucial part of deep space exploration missions.

REFERENCES

[1] Xu Hue. Research on data compression algorithm in real-time database [D]. Zhejiang University, 2006.

[2] Wang HI. LZW dictionary compression improved algorithm research and FPGA hardware implementation [D]. Nanjing Normal University, 2012.

[3] Chen Changsha. Research and Design of Data Compression Algorithm [D]. Central South University, 2010.