RESEARCH ARTICLE OPEN ACCESS

Arithmetic Coding for Lossless Data Compression – A Review

Ezhilarasu P [1], Krishnaraj N [2], Dhiyanesh B [3]
Associate Professor [1], Assistant Professor [3]
Department of Computer Science and Engineering
Hindusthan College of Engineering and Technology
Coimbatore
Head of the Department [2]
Department of Information Technology
Sree Sastha Institute of Engineering and Technology
Chennai
Tamil Nadu – India

ABSTRACT

In this paper, Arithmetic Coding data compression technique reviewed. Initially, arithmetic encoding performed for the taken input. Then decoding for the obtained result done. It regenerates the original uncompressed input data. Its compression ratio, space savings, and average bits also calculated.

Keywords:- Arithmetic Coding, Compression, Encoding, Decoding.

I. INTRODUCTION

Data compression defined as the representation of data in such a way that, the storage area needed for target data is less than that of the size of the input data. The decompression technique regenerates the source data. After decompression, if there is some loss of data, then the compression called as lossy compression. If none of the data missed, then the compression named as lossless compression. The Arithmetic coding comes under lossless compression. Each compression technique looks for two important aspects. Those are complexity in terms of space along with time.

Arithmetic coding generates variable length codes. It bypasses traditional methods of replacing input characters by specific code, like code words. It uses the combination of both integers and floating-point numbers. The integers used initially to represent two limits. Those are high limit one and low limit zero. Then in the subsequent steps these limits change into floating-point numbers. The floating-point numbers used to represent the input. The output of an arithmetic encoding is the collection of bits derived from the floating-point number. The binary converted into fractional number, then regenerates the input in arithmetic decoding.

II. RELATED WORK

Shannon [1948] showed that it was possible to generate better compression code for the probability model. He produced minimum average bits per symbol for the given input [1]. Fano [1949] also provided optimal code by working on data compression [2]. Huffman, the student of Fano also worked on producing optimal code better than that of Shannon-Fano coding. The Shannon-Fano coding is a top-down approach. Huffman [1952] used the bottom-up approach to producing better optimal code than the work of his master [3]. The significant advantage of arithmetic coding is its flexibility and optimality. In most cases, Huffman coding produces very nearly optimal code [4, 5, 6, and 7]. The main limitation of arithmetic coding is its slowness. Huffman coding and Lempel-Ziv coding are faster [8, 9] than arithmetic coding. The approximation technique used to increase the speed of the coding [10, 11, 12, and 13].

III. ARITHMETIC ENCODING

In the field of data compression, arithmetic coding is entropy encoding. The floating-point number calculated by the characters probability. In each step of arithmetic coding, the value of the lower limit increases or remains the same. Whereas, for the upper bound, the value decreases or remains the same. So lower limit value always greater than or equal to previous lower limit value. The top limit value, always less than or equal to the previous upper bound value.

A. ALGORITHM

1. Get the input.
2. Read the information, character by character.
3. Identify unique characters and its occurrences.
4. Find the probability of each character.
5. Initialize the high limit = one, low limit = zero.
6. Find the low limit and high limit probability for each unique character.
7. Read the character from the left to right.
8. Find low limit by using the equation 1,

Low limit = previous low limit
+ (Previous high limit – previous low limit)
* Low limit probability of the taken character (1)

If the low limit probability of the chosen character is zero, then low limit = previous low limit.

9. Find high limit by using the equation 2,

High limit = previous high limit -
(Previous high limit – previous low limit)
* (1-high limit probability of the taken character) (2)

If a high limit probability of the chosen character is one, then high limit = previous high limit.

10. Apply step seven, eight, and nine recursively to the remaining characters.
11. Find the average of low limit and high limit.
12. Convert the output fractional number into binary.

B. EXAMPLE

If in a message (M), whose length is ten we have four, unique characters (m1, m2, m3, m4) with occurrences are 4, 2, 3, and 1. The probability (P) of each unique character given as (p1, p2, p3, p4) given in the equation 3.

Unique character probability (P) =
Character (Symbol) occurrences /Message length (3)

The probability of the unique characters (m1, m2, m3, m4) calculated by (p1, p2, p3, p4) using the equation 3. It is given in the coding table as provided in the table I.

TABLE I. CODING TABLE

CHARACTER	m1	m2	m3	m4
OCCURRENCE	4	2	3	1
PROBABILITY	0.4	0.2	0.3	0.1

The table used to derive low limit probability and high limit probability for each unique character as shown in Table II.

TABLE II. CODING TABLE WITH LOW AND HIGH LIMIT PROBABILITY

S.NO	UNIQUE CHARACTER	LOW LIMIT PROBABILITY	HIGH LIMIT PROBABILITY
1.	m1	0	0.4
2.	m2	0.4	0.6
3.	m3	0.6	0.9
4.	m4	0.9	1

The table used to derive the fractional number for the given input message (M).

If the message is “arasu” with each character probability given as in the table III.

C. IMPLEMENTATION

1. Get the input (“arasu”).
2. Read the input character by character (‘a’,’r’,’a’,’s’,’u’).
3. Identify unique character and its occurrences (‘a-1’,’r-1’,’a-2’,’s-1’,’u-1’).

4. Find the probability of each unique character, as shown in the table III.

TABLE III. UNIQUE CHARACTER PROBABILITY

S. NO	CHARACTER	PROBABILITY
1	‘a’	0.4
2	‘r’	0.2
3	‘s’	0.2
4	‘u’	0.2

5. Initialize the high limit = one, low limit = zero.
6. Find the low limit and high limit probability for each unique character, as shown in the table IV.

TABLE IV. CODING TABLE WITH LOW AND HIGH LIMIT PROBABILITY

S. NO	UNIQUE CHARACTER	LOW LIMIT PROBABILITY	HIGH LIMIT PROBABILITY
1.	a	0	0.4
2.	r	0.4	0.6
3.	s	0.6	0.8
4.	u	0.8	1

7. Read the character from the left to right (‘a’).
8. Find low limit by using the equation 1,
Low limit = previous low limit + (Previous high limit – previous low limit) * Low limit probability of the taken character
Here the low limit probability of the character ‘a’ is zero. Hence, Low limit = zero.
9. Find high limit by using the equation 2,
High limit = previous high limit - (Previous high limit – previous low limit) * (1-high limit probability of the taken character)
= 1 – (1 – 0) * (1 – 0.4)
= 1 – 1 * 0.6
= 1 – 0.6
= 0.4
10. Apply step seven, eight, and nine recursively to the remaining characters.
7. Read the character from the left to right (‘r’).
8. Find low limit by using the equation 1,
Low limit = previous low limit + (Previous high limit – previous low limit) * Low limit probability of the taken character
= 0 + (0.4- 0) * 0.4
= 0 + 0.4 * 0.4
= 0 + 0.16
= 0.16
9. Find high limit by using the equation 2,
High limit = previous high limit - (Previous high limit – previous low limit) * (1-high limit probability of the taken character)
= 0.4 – (0.4 – 0) * (1 – 0.6)
= 0.4 – 0.4 * 0.4
= 0.4 – 0.16
= 0.24
10. Apply step seven, eight, and nine recursively to the remaining characters.
7. Read the character from the left to right (‘a’).
8. Find low limit by using the equation 1,
Low limit = previous low limit + (Previous high limit – previous low limit) * low limit probability of the taken character
Here the low limit probability of the character ‘a’ is 0. Hence, Low limit = 0.16.
9. Find high limit by using the equation 2,
High limit = previous high limit - (Previous high limit – previous low limit) * (1-high limit probability of the taken character)
= 0.24 – (0.24 – 0.16) * (1 – 0.4)
= 0.24 – 0.08 * 0.6
= 0.24 – 0.048
= 0.192
10. Apply step seven, eight, and nine recursively to the remaining characters.
7. Read the character from the left to right (‘s’).
8. Find low limit by using the equation 1,
Low limit = previous low limit + (Previous high limit – previous low limit) * Low limit probability of the taken character
= 0.16 + (0.192 – 0.16) * 0.6
= 0.16 + 0.032 * 0.6
= 0.16 + 0.0192
= 0.1792
9. Find high limit by using the equation 2,
High limit = previous high limit - (Previous high limit – previous low limit) * (1-high limit probability of the taken character)
= 0.192 – (0.192 – 0.16) * (1-0.8)
= 0.192 – 0.032 * 0.2
= 0.192 – 0.0064
= 0.1856
10. Apply step seven, eight, and nine recursively to the remaining characters.
7. Read the character from the left to right (‘u’).
8. Find low limit by using the equation 1,
Low limit = previous low limit +

(Previous high limit – previous low limit)
* Low limit probability of the taken character
= 0.1792 + (0.1856 – 0.1792) * 0.8
= 0.1792 + 0.0064 * 0.8
= 0.1792 + 0.00512
= 0.18432

9. Find high limit by using the equation 2,
High limit = previous high limit -
(Previous high limit – previous low limit)
* (1-high limit probability of the taken character)

Here, the high limit probability of the chosen character is 1. So high limit = previous high limit.

High limit = 0.1856

10. Apply step seven, eight, and nine recursively to the remaining characters. The remaining input characters are nil.

11. Find the average of low limit and high limit.
Average = (0.18432 + 0.1856) / 2
= 0.18496

12. Convert the output fractional number into binary.
0.18496 = 0.0010111101011001100010 (eliminate prefix 0.)
= 0010111101011001100010

The whole encoding process, as in the Table V.

TABLE V. ARITHMETIC ENCODING

S. NO	UNIQUE CHARACTER	PROBABILITY	LOW LIMIT PROBABILITY	HIGH LIMIT PROBABILITY	LOW LIMIT	HIGH LIMIT
1	'a'	0.4	0	0.4	0	0.4
2	'r'	0.2	0.4	0.6	0.16	0.24
3	'a'	0.4	0	0.4	0.16	0.192
4	's'	0.2	0.6	0.8	0.1792	0.1856
5	'u'	0.2	0.8	1	0.18432	0.1856

0010111101011001100010

The total number of bits needed = 22 bits.
The size of the input as uncompressed
= 5 * 8
= 40 bits

IV. ARITHMETIC DECODING

The arithmetic decoding regenerates the original input.

A. *ALGORITHM*

1. Convert the binary to fractional number.
2. Initialize high limit as one and low limit as zero.
3. Divide the range using the probability of each unique character.
4. Check the range of the fractional number.
5. Write the corresponding character.
6. Find the average of the low limit and high limit. If the average and fractional number both are same then stop the process.
7. Apply step 3 – 6 recursively.

B. *IMPLEMENTATION*

1. Convert the binary to the fractional number (0.18496).
2. Initialize high limit as one and low limit as zero.
3. Divide the range using probability of each unique character (0 – 0.4 for 'a', 0.4 – 0.6 for 'r', 0.6 – 0.8 for 's', and 0.8 – 1 for 'u').
4. Check the range of the fractional number (0.18496 fits the range 0 - 0.4).
5. Write the corresponding character ('a').
6. Find the average of the low limit and high limit. If the average and fractional number both are same then stop the process. Average = 0.4/2 = 0.2 not equal to 0.18496.
7. Apply step 3 – 6 recursively.
3. Divide the range using probability of each unique character (0 – 0.16 for 'a' 0.16 – 0.24 for 'r', 0.24 – 0.32 for 's', and 0.32 – 0.4 for 'u').
4. Check the range of the fractional number (0.18496 fits the range 0.16 - 0.24).
5. Write the corresponding character ('r').
6. Find the average of the low limit and high limit. If the average and fractional number both are same then stop the process. Average = 0.4/2 = 0.2 not equal to 0.18496.
7. Apply step 3 – 6 recursively.

3. Divide the range using probability of each unique character (0.16 – 0.192 for ‘a’, 0.192 – 0.208 for ‘r’, 0.208 – 0.224 for ‘s’, and 0.224 – 0.24 for ‘u’).
4. Check the range of the fractional number (0.18496 fits the range 0.16 - 0.192).
5. Write the corresponding character (‘a’).
6. Find the average of the low limit and high limit. If the average and fractional number both are same then stop the process. Average = 0.352/2 = 0.176 not equal to 0.18496.
7. Apply step 3 – 6 recursively.
3. Divide the range using probability of each unique character (0.160 – 0.1728 for ‘a’, 0.1728 – 0.1792 for ‘r’, 0.1792 – 0.1856 for ‘s’, and 0.1856 – 0.192 for ‘u’).
4. Check the range of the fractional number (0.18496 fits the range 0.1792 - 0.1856).
5. Write the corresponding character (‘s’).
6. Find the average of the low limit and high limit. If the average and fractional number both are same then stop the process. Average = 0.3648/2 = 0.1824 not equal to 0.18496.
7. Apply step 3 – 6 recursively.
3. Divide the range using probability of each unique character (0.1792 – 0.18176 for ‘a’, 0.18176 – 0.18304 for ‘r’, 0.18304 – 0.18432 for ‘s’, and 0.18432 – 0.1856 for ‘u’).
4. Check the range of the fractional number (0.18496 fits the range 0.18432 – 0.1856).
5. Write the corresponding character (‘u’).
6. Find the average of the low limit and high limit. If the average and fractional number both are same then stop the process. Average = 0.36992/2 = 0.18496 equal to 0.18496. Hence, stop the process.

The entire decoding process, as in the table VI.

TABLE VI. ARITHMETIC DECODING

S.NO	FRACTIONAL NUMBER	RANGE OF UNIQUE CHARACTER				OUTPUT CHARACTER	AVERAGE
		‘a’	‘r’	‘s’			‘a’
1	0.18496	0-0.4	0.4-0.6	0.6-0.8	1	0.18496	0-0.4
2	0.18496	0-0.16	0.16-0.24	0.24-0.32	0.32-0.4	‘r’	0.2
3	0.18496	0.16-0.192	0.192-0.208	0.208-0.224	0.224-0.24	‘a’	0.176
4	0.18496	0.16-0.1728	0.1728-0.1792	0.1792-0.1856	0.1856-0.1920	‘s’	0.1824
5	0.18496	0.1792-0.18176	0.18176-0.18304	0.18304-0.18432	0.18432-0.1856	‘u’	0.18496

V. RESULT AND DISCUSSION

The compression ratio, space savings and average bits calculated for the examples are

Compression ratio = 40/22
= 20:11
=1.82:1

Space savings =1-(22/40)
= 1-(11/20)
= 1-0.55
= 0.45
= 45%

Average bits = 22/5
= 4.4 bits per character

VI. CONCLUSION

The arithmetic coding is an innovative compression technique. It represents entire message by using the floating-point number as compared to code words. The obtained results depict that arithmetic coding gives better compression ratio, space savings, and average bits per character.

REFERENCE

[1] C. E. Shannon, “A Mathematical Theory of Communication”, Bell System Technical Journal, Volume 27, pp. 398-403, 1948.

[2] R.M. Fano, “The transmission of information”, Technical Report 65, Research Laboratory of Electronics, M.I.T, Cambridge, Mass, 1949.

[3] D.A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes", Proceedings of the I.R.E, pp. 1098–1102, 1952.

[4] A. C. Blumer & R. J. McEliece, “The Renyi Redundancy of Generalized Huffman Codes," IEEE Trans. Inform. Theory, IT-34, pp.1242-1249, 1988.

[5] R. M. Capocelli, R. Giancarlo & I. J. Taneja, “Bounds on the Redundancy of Huffman Codes," IEEE Trans. Inform. Theory, IT-32, pp. 854-857, 1986.

[6] R. G. Gallager, “Variations on a Theme by Huffman," IEEE Trans. Inform. Theory, IT-24, pp. 668-674,1978.

[7] S. Parker, “Conditions for the Optimality of the Huffman Algorithm," SIAM J. Comput. 9, pp., 470-489, 1980.

[8] J. Ziv & A. Lempel, “A Universal Algorithm for Sequential Data Compression", IEEE Trans. Inform. Theory, IT-23, pp.337-343, 1977.

[9] J. Ziv & A. Lempel, “Compression of Individual Sequences via Variable-Rate Coding", IEEE Trans. Inform. Theory, IT-24, pp. 530-536, 1978.

[10] Chevion, E. D. Karnin & E. Walach, “High Efficiency, Multiplication Free Approximation of Arithmetic Coding," in Proc. Data Compression Conference, J.A. Storer & J. H. Reif, eds., Snowbird, Utah, Apr. 8-11, pp. 43-52, 1991.

[11] G. G. Langdon, “Probabilistic and Q-Coder Algorithms for Binary Source Adaptation," in Proc. Data Compression Conference, J. A. Storer & J. H. Reif, eds., Snowbird, Utah, Apr. 8-11, pp. 13-22, 1991.

[12] G. G. Langdon & J. Rissanen, “Compression of Black-White Images with Arithmetic Coding," IEEE Trans. Comm. COM-29, pp. 858-867, 1981.

[13] W. B. Pennebaker, J. L. Mitchell, G. G. Langdon & R. B. Arps, “An Overview of the Basic Principles of the Q-Coder Adaptive Binary Arithmetic Coder," IBM J.Res. Develop. 32, pp. 717-726, 1988.