A REVIEW OF LOSSLESS DATA COMPRESSION ALGORITHMS

REVIEW ARTICLE

Aditya Meharia
School of Computer Engineering
Kalinga Institute of Industrial Technology
Bhubaneswar, India 751024
adityameharia14@gmail.com

Junaid H. Rahim
School of Computer Engineering
Kalinga Institute of Industrial Technology
Bhubaneswar, India 751024
junaidrahim5a@gmail.com

April 11, 2020

ABSTRACT

This work aims to provide an introduction to the domain of Data Compression in Information Theory and a comprehensive review of the existing literature in the field of algorithms for Lossless Data Compression. We identify and discuss the potential opportunities, barriers and the future scope of the field. We also review data compression methods used for text, image, video and audio data.

***K*eywords** Data Compression · Algorithms · Lossless Data Compression · Information Theory

1 Introduction

A rapid growth in modern communication technology led an explosion in the amount of data we transmit and store. Large files consume significant resources for transmission as well as storage. Due to this exponential increase in the size of the data we transmit, researchers developed algorithms that can be used to compress the data to save storage space as well as transmission time. Data Compression is a process by which we encode the input data into a representation that occupies fewer bits than the original input. This encoded representation is transmitted and decoded back to the original form at the destination. Data compression algorithms are broadly classified into two classes viz **Lossless Compression** and **Lossy Compression** algorithms. We will be only covering Lossless Data Compression algorithms in this review article.

1.1 History of Data Compression

The field of data compression gained huge significance in the 1970s after the surge in the usage of the internet. The need to reduce transmission time pushed computer scientists to find new ways to compress information. Although, the very earliest form of compression was Morse Code, invented in 1838, in Morse code the letters 'e' and 't' from the english language were given shorter codes as they have a high probability of occurrence.

Later with the advent of mainframe computers, Calude Shanon and Robert Fano invented Shanon-Fano coding in 1949[1], the algorithm assigns shorter codes to symbols with high probability resulting in a shorter way to represent the data. In 1952, David Huffman, one of the students of Robert Fano at MIT studying information theory took the option to write a term paper when given a choice between taking a final exam or writing a paper. Huffman was interested in finding the most efficient way to assign prefix codes to a set of symbols, after months of work Huffman published Huffman Coding in his paper "A Method for the Construction of Minimum-Redundancy Codes"[2], Huffman coding was an improvement over Shanon-Fano coding in terms of efficiency as it assured the assignment of the shortest possible codes to the given symbols. The early implementations of Shanon-Fano coding and Huffman coding were done using hardcoded codes, later in the 1970s, software compression was implemented and Huffman Codes were dynamically generated depending on the input data.

In 1977, Abraham Lempel and Jacob Ziv published[3] their groundbreaking LZ77 algorithm and later the LZ78 algorithm, these algorithms used a dictionary to compress data. The popular UNIX operating system used a compression utility based on LZW which was a slight modification of the LZ78 algorithm. Later the UNIX community adopted the

DEFLATE based gzip and Burrows-Wheeler transform based bzip2 formats mostly due to their open source nature[4]. It was a beneficial decision in the long run as gzip and bzip2 have consistently given higher compression ratios compared to the LZW format.

In 1989, Phil Katz released the PKZIP format, later in 1993 Katz updated the format and named it PKZIP 2.0, he based it on the DEFLATE algorithm, the .zip format used so extensively in today's day is based on the PKZIP 2.0 format. ZIP and other DEFLATE based formats were extremely popular till the mid 1990s when new and improved formats began to emerge. In 1993, Eugene Roshal released his WinRAR utility which uses the proprietary RAR format. The RAR format is one of the most used formats to compress data and share it via the internet. In 1999, UNIX adopted the 7-zip or the .7z format, this was the first one capable enough to challenge the dominance of the .zip and .rar formats as .7z was not limited to just one compression algorithm, but could instead choose any of bzip2, LZMA, LAMA2 and PPMd algorithms among others.

1.2 Overview of Lossless Data Compression Techniques

Lossless Compression algorithms are a class of algorithms that can reproduce the original content from the encoded respresentation without any loss of information, the data before compression and after decompression is exactly the same. Lossless compression is used in a variety of fields where it is important that the original and decompressed information be the same. The GNU tool gzip uses lossless algorithms for the ZIP file format.

Lossless compression algorithms usually have a two step procedure.

1. A statistical model of the input data is generated. This usually assigns a probability of occurrence to pieces of input data. For example, if the input data is piece of text, then the model would be the probabilities of occurance of each alphabet
2. A coding system uses this model to map the data in a way that the pieces with high probability of occurrence are assigned a shorter code than those with a low probability of occurance

The probabalistic model is usually generated in two ways, a static way and an adaptive/dynamic way. In the static approach, the data is analysed and the probability model is generated before starting the encoding procedure, this is a modular and simple approach but doesn't perform well for heterogeneous data since the approach forces the use of a single model for the all the data. In the dynamic method, the model is updated while compressing the data. The encoder and decoder start with a trivial model in the initial state, thus performs poorly on initial data, but as the model adapts to the data, the performance improves. Most efficient compression techniques usually employ an adaptive model. There are various ways to achieve losseless compression namely Run Length Encoding (RLE), Lossless predictive coding (LPC), Entropy coding and Arithmetic coding etc.[5][6]

2 Prefix Codes and Entropy

A prefix code is a "code" system in which the codes given to each character is not the prefix of the code given to any other character i.e. it follows the prefix property [7]. For example $\{2, 42, 12\}$ is an example of a prefix system whereas $\{2, 42, 12, 21\}$ is not because '2' is the prefix of '21'.

Prefix codes are also known as prefix-free codes, prefix condition codes and instantaneous codes. They are uniquely decodable codes that is no two codes will have the same value on decoding. Prefix codes can be both fixed length and of variable length. It does not require between words to seperate them. Variable length preifx codes have been used extensibly in Huffman and Shannon coding and are still used in modern compression algorithms along with arithmetic coding.

Entropy denotes the randomness of the data that you are passing as input to the compression algorithm. That means the more random the text i.e. higher entropy is, the lesser you can compress it. It represents an absolute limit on the best possible lossless compression of any communication: treating messages to be encoded as a sequence of independent and identically distributed random variables. Shannon's source coding theorem shows that, the average length of the shortest possible representation to encode the messages in a given alphabet is expressed as follows

Given a random variable X, with possible outcomes x_i, each with probability $P_X(x_i)$, the entropy $H(X)$ of X, where b is the base of the logarithm is as follows:

$$H(X) = -\sum_{i=1}^{n} P_X(x_i) \log_b P_X(x_i) = \sum_{i=1}^{n} P_X(x_i) I_X(x_i) = E[I_X]$$

3 Shanon Coding

It is named after its creator **Claude Shanon**, the technique was used to prove Shanon's noiseless coding theorem in his 1948 article "A Mathematical Theory of Communication"[1]. Even though being suboptimal, the method was a first of its kind. This method is credited to have given rise to the entire field of Information Theory. Some of the most efficient compression algorithms today are usually an exntension of shanon's method

Shanon Coding is a method to generate prefix codes for a given piece of data. It is done using the occurrence probabilities of the pieces of data. First the probabilities p_i are arranged in descending order, then each piece is assigned a code which is the first l_i digits of binary representation of the cumulative probability till that piece of data.

Given that the probability of occurrence is p_i, the cumulative probability is expressed as

$$\sum_{k=0}^{i-1} p_k$$

where $l_i = \lceil \log_2 p_i \rceil$

It is a suboptimal algorithm, it does not give the lowest possible code word length.

The following is an example of assigning prefix codes to compress the string "lossless data compression"

i	a_i	p_i	$p_c = \sum_{k=0}^{i-1} p_k$	Binary Representation	$l_i = \lceil \log_2 p_i \rceil$	code
0	s	0.24	0.00	0.00000000...	2	00
1	o	0.12	0.24	0.00111101...	3	001
2	e	0.08	0.36	0.01011100...	3	010
3	<space>	0.08	0.44	0.01110000...	3	011
4	a	0.08	0.52	0.10000101...	3	100
5	l	0.08	0.60	0.10011001...	3	100
6	i	0.04	0.68	0.10101110...	4	1010
7	d	0.04	0.72	0.10111000...	4	1011
8	t	0.04	0.76	0.11000010...	4	1100
9	c	0.04	0.80	0.11001100...	4	1100
10	m	0.04	0.84	0.11010111...	4	1101
11	r	0.04	0.88	0.11100001...	4	1110
12	p	0.04	0.92	0.11101011...	4	1110
13	n	0.04	0.96	0.11110101...	4	1111

4 Huffman Coding

Named after its creator David A. Huffman. Although C.E. Shannon [1] and R.M.Fano [8] developed ensemble coding procedures to prove that the average number of binary digits required per message approaches from above the average amount of information per message,it was not optimum. Kraft [9] had derived a coding method which gives an average code length as close as possible to the ideal when the ensemble contains a finite number of members. However Huffman was able to derive a definte procedure for this. The output of the Huffman table given us a prefix-variable code table which consists of the source symbol and the encoded symbol.

Like Shannon coding, Huffman algorithm also tries to minimize the entropy by assisgning shortest codes to the charcters which occur most frequently. The algorithm works by creating a binary tree (using a min heap) which can have either leaf nodes or internal nodes. The leaf node contains the weight and the symbol whereas the internal nodes consists of weight and links to the two child nodes. The bit '0' is used to reprsent the left child of an internal nodes whereas the bit '1' is used to represent the right child.

Steps to build a Huffman Tree:-

1. The process begins by traversing the input string and finding all the unique charcters along with their frequencies.
2. Then create a leaf node for each symbol and their frequencies(weight) and add build a min heap of all the leaf nodes.

3. Take the two nodes with the minimum weight and create a new internal node with the weight equal to the sum of the frequencies of the 2 nodes.Make the first extracted child as the left node and the second extracted child as the right node and add it to the min heap.

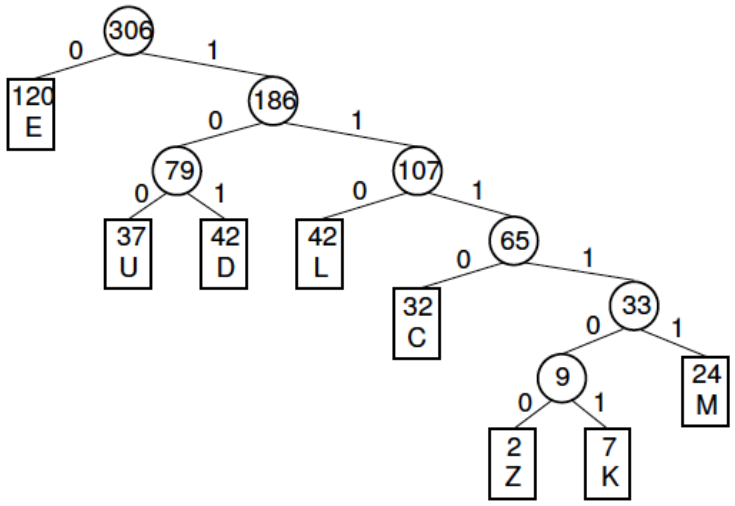
4. Repeat the steps until the heap contains only one node.

Letter freqency table

Letter	Z	K	M	C	U	D	L	E
Frequency	2	7	24	32	37	42	42	120

Huffman Code

Letter	Freq	Code	Bits
E	120	0	1
D	42	101	3
L	42	110	3
U	37	100	3
C	32	1110	4
M	24	11111	5
K	7	111101	6
Z	2	111100	6



Decoding the Huffman tree is very simple, traverse the tree node by node as each bit is read from the input stream (reaching a leaf node necessarily terminates the search for that particular byte value).

For example, decoding an encoded string can be done by looking at the bits in the coded string from left to right until a letter decoded. 10100101 $\Rightarrow$ DEED

5 Lempel-Ziv(lz) Compression Methods

6 Run Length Coding

7 Prediction by Partial Matching

8 Arithmetic Coding

9 Deflate

10 Grammar Based Compression

11 Current Research Work

12 Future Scope

13 Conclusion

References

[1] Claude E Shannon. A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423, 1948.

[2] David A Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

[3] Jacob Ziv and Abraham Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on information theory*, 23(3):337–343, 1977.

[4] Michael Burrows and David J Wheeler. A block-sorting lossless data compression algorithm. 1994.

[5] PM Parekar and SS Thakare. Lossless data compression algorithm–a review. *International Journal of Computer Science & Information Technologies*, 5(1), 2014.

[6] P Yellamma and Narasimham Challa. Performance analysis of different data compression techniques on text file. *International Journal of Engineering Research & Technology (IJERT)*, 1(8):1–6, 2012.

[7] Jean Berstel and Dominique Perrin. *Theory of codes*. Academic Press, 1985.

[8] Robert M Fano. *The transmission of information*. Massachusetts Institute of Technology, Research Laboratory of Electronics . . . , 1949.

[9] Leon Gordon Kraft. *A device for quantizing, grouping, and coding amplitude-modulated pulses*. PhD thesis, Massachusetts Institute of Technology, 1949.