Enhancement of Data Compression Using Incremental Encoding

Ajit Singh and Yogita Bhatnagar

Abstract— Now a day's compression is becoming more popular, because it helps to reduce the size of data from its original size of the data. This paper describes the two phase encoding technique which compresses the sorted data more efficiently. The research paper provides a way to enhance the compression technique by merging RLE compression algorithm and incremental compression algorithm. In first phase the data is compressed by RLE (Run length encoding) algorithm that compresses the frequent occur data bits by short bits. In the second phase incremental compression algorithm stores the prefix of previous symbol from the current symbol and replaces with integer value. The proposed technique increases the compression rate as from RLE compression algorithm and incremental compression algorithm. In future the proposed mechanism will be very beneficial for compression large amount of data.

Index Terms— Combination of Incremental & RLE encoding, Compression, Compression technique, Enhancement of Incremental algorithm, Incremental encoding, Lossless compression, Lossy compression, Run length encoding

1 INTRODUCTION

Data compression is an art of reducing the size of original data and data compression stores the same amount of data in few bits.[1] Data compression is also referred to as encoding, encoding generally encompasses the special representation of data which satisfies the need. This Paper specifies the study of efficient encoding and develops an algorithm to encode the data into as few bits as possible. The primary objective of data compression is to minimize the amount of data to be transmitted. The theme of this paper is to present and analyze a data compression technique.

Compression is necessary. It helps to reduce the consumption of expensive resources, such as hard disk space or transmission bandwidth. Compressed data can be decompressed when required. [2]

Many data processing applications require storage of large volumes of data, the number of such applications is constantly increasing. At the same time, the communication networks are having in massive transfer of data through communication channels. Compressed data is stored or transmitted that reduces storage and communication costs. When reduced data is transmitted through links, the effect will increase the capacity of the communication links. Similarly the capacity of the storage medium is double, by compressing a file to half of its original size. This becomes viable to store the data at a higher that reduce the load on the input/output channels of the computer system. [3]

2 COMPRESSION TECHNIQUES

Compression techniques are classified into two types of compression algorithm classes.

- Lossy compression algorithms
- Lossless compression algorithms

The compression algorithms take input Data A and compressed that input data to Data A Compressed and there are also decompressed algorithm for each compression algorithm to regenerate the data. Decision for which type of class of algorithm to use for compression & decompression is depends on the type of data and also the requirement of the user.

- *Dr. Ajit Singh is presently working as Chairperson in School of Engineering & Sciences in BPSMV, Khanpur Kalan (Sonipat) and also having additional charge as Director, Computer Centre (UGC). He posses qualifications of B.Tech, M.Tech, PhD (p). He is a member of BOG (Board of Governors) of Haryana State Counseling Society, Panchkula and also member of academic council in the University. He published approximate 20 papers in National/ International journals and conferences and holds a teaching experience of approximate 10 years. He holds the membership of Internal Quality Assurance cell, UG-BOS & PG-BOS and the NSS advisory committee. He is also an associate member of CSI & IETE. His research interests are in Network Security, Computer Architecture and Data Structure. E-mail: ghanghas_ajit@rediffmail.com*
- *Mrs. Yogita Bhatnagar has completed her B.E degree in Information Technology from Vaish College of Engineering, Rohtak, Maharshi Dayanand University (MDU), India in the year 2008, and she is pursuing M.Tech in Information Technology from Bansathali University, Banasthali since June 2010. Currently she is doing internship from B.P.S.M.V Khanpur Kalan, Sonipat. She has one year working experience as Business Analyst in an organization. She has presented papers in International and National Conferences. Her research interests are in Network Security, Software Engineering and Data Warehousing. PH-9996325336. E-mail: y.yogita.1986@gmail.com*

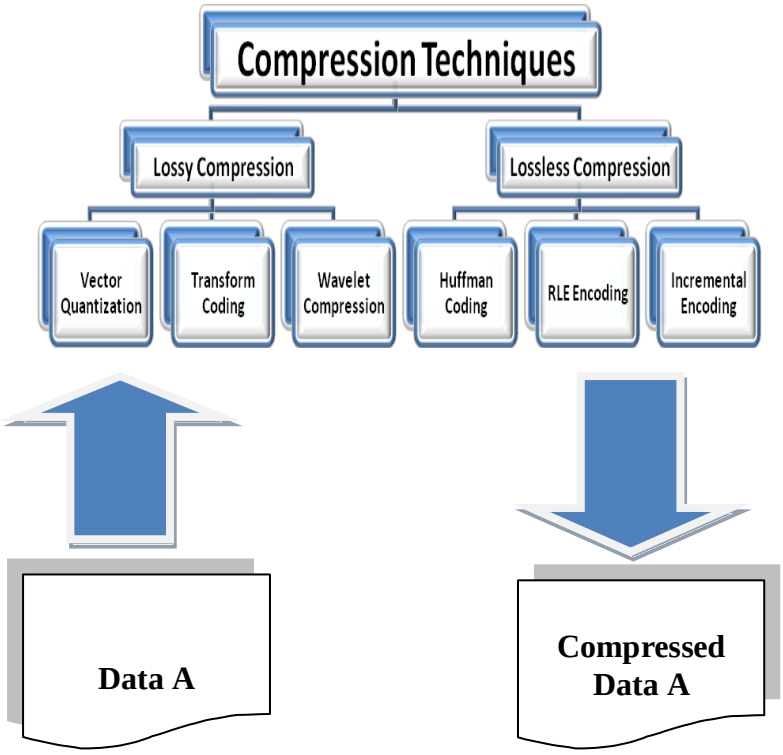

Figure 1: Compression techniques

2.1 Lossless Compression

In lossless compression there is no loss of information. If data compression technique is lossless data compression technique then the original data can be easily regenerate from the compressed data.
For example: - lossless compression is very useful in text compression.
Because reconstruction of text message as the original message is very important, a small change in the message can change the meaning of the message. For example "DO NOT SPIT HERE" and "DO SPIT HERE".
Lossless data compression is used in many applications. For example, it is used in the ZIP file format, also in the Abraham Lempel algorithm. LZ77 and LZW are lossless data compression techniques.
In the circumstances where we require data after decompression exactly as the original data, we use lossless compression technique. [4]
There are several circumstances where we does not require data exactly same as the original data so at that time we can proceed for lossy compression.

2.2 Lossy Compression

As its name suggest there is a loss of information during compression but it increases compression rate. And the compressed data by using this technique does not obtained as the original data. For recover that loss of information we use much higher compression ratio for compressing the data.
For example: there are some areas where exact original data is not required. During transmitting some speech and video, the data are not required as exact as original. They decompress according to the quality of the data required. This type of compression is used where any loss of information is affordable.

Lossy compression is used to compress multimedia data (audio, video and still images) such as streaming media and internet telephony. In comparison the lossless compression is required for text and data files, such as bank records and text articles where exact data is required after decompression. [4]

User can use different data compression techniques according to their requirement. If user needs more compression and can do compromise with loss of data then lossy compression is well suited to him and if user can't compromise with loss of data then Lossless compression is the best option for him.

3 Run Length Encoding

Run-length encoding (RLE) is a very popular lossless data compression technique which follows the idea of compressing the long sequence of same characters by short values. Run length encoding is very good compression technique. It encodes the same data ('aaaaa') and count number of repetitions in that data. [1][2]
The idea of RLE is very simple, suppose we have a string 'yyyyy'. The output of RLE is 5'y', 5 meaning that there are 5 bytes of same type of data. It reduces the physical size of consecutive string of characters.

3.1 Implementation

RLE implementation is very easy, if there are two bytes, then both of have equal output and then count how many bytes are equal, then output that message value and continue encoding, then remove the repeated byte values .The value can't be greater than 255 because we are using a byte to represent that message. If the message bytes are not equal, then output the first message, make the second message the first, and get the next byte as a second, and start again.

The message is divided into 2 parts, first part is the counter, that represents that the number of values in the message. The size would be of 1 to 128 or 256 characters and the counter having values number of character minus 1. Second part of the message is the message itself which has the value from 0 to 255 characters.
For example:-
AAAAAAAAAAAAAA
Our uncompressed message will take 15 bytes to store but after implementation of RLE Compression algorithm our message will take 2 bytes memory to store that message value.

AAAAAAAAAAAAAAA -> 15A
Here the counter value is 15.
Consider new message: AAAAAAcccdddddy
The new message is encoded in 4 new 2 bytes packets by using RLE compression algorithm.
6A3c5d1y

By using RLE encoding the 15 byte string message is compressed to 8 bytes. RLE provides almost 67% compression ratio.

RLE schemes are simple and efficient. Its compression depends on the type of the data to be compressed. RLE is very fast and efficient algorithm that uses different run length encoding schemes. As we have seen various examples to compress string message by using RLE encoding algorithm. In many situations RLE implements according to the type of data.

- RLE is easy and fast encoding algorithm. It does not require much CPU POWER.
- RLE implements only when repetitive data is of large amount.
- RLE algorithm implements different schemes depending on the type of the data.

4 Incremental Encoding Algorithm

Incremental algorithm stores the common prefixes or suffixes with their lengths to remove the reoccurrence. That's why it is also known as front compression and back compression .The major property of incremental algorithm is that suitable for sorted data. For e.g. list of words from a dictionary and spread sheets.
Following is the list of such words:

Input	Common prefix	Compressed output
Abcd	No preceding word	0 abcd
abcophyta	`abc`	3 ophyta
abcopgh	`abcop`	5 gh
Tab	No common prefix	0 tab
Tabbed	`Tab`	3 bed
Tabbing	`Tabb`	4 ing
Tabit	`Tab`	3 it

Table 1: Results of Incremental algorithm

The incremental encoding stores the common prefix length that varies from application to application. This technique stores the value as a single byte which stores only the change in the common prefix length and various universal codes. It may be combined with other general lossless data compression techniques such as entropy encoding and dictionary coders to compress the remaining suffixes.

5 Proposed Technique

The propose technique first apply the RLE compression algorithm on data and then apply the incremental compression algorithm on the output of RLE algorithm. We are using "." as escape character to differentiate between RLE compressed data and Incremental compressed data.

5.1 Comparison Analysis

Let's take an example to understand the proposed technique and its advantages in terms of compression ratio with RLE encoding & incremental encoding. "." is used to differentiate data between Incremental and RLE algorithm. In this example we are taking 3 words as input and compressing the input words using RLE, Incremental & our proposed technique.
Below is the comparison table of the algorithm:

Table 2: Comparison analysis table

Input Word	RLE output	Incremental Output	Proposed Technique output
AAAAABBBCCCDDE	5A3B3CDDE	AAAAABBBCCCDDE	5A3B3CDDE
AAAAABBBCD	5A3BCD	8CD	9.D
AAAAABBCCCDD	5ABB3CDD	7CCCDD	7.3CDD
Takes 36 bytes	Takes 23 bytes	Takes 23 bytes	Takes 18 bytes
Compression%	**33%**	**33%**	**50%**

Uncompressed data takes 36 bytes and after applying RLE encoding alone it will take 23 bytes and after applying Incremental encoding alone it will take 23 bytes. But by using our proposed technique data is taking 18 bytes .Thus we got **compression ratio of 50%** by using our proposed technique.

5.2 Implementation

We have implemented the proposed technique in two phases. Following are the descriptions of two phases:

First Phase: We are applying RLE encoding technique to compress the input data.
Second Phase: We are applying Incremental encoding technique on the output of RLE encoding technique.
The final result is more compressed than the RLE encoding & Incremental encoding. Also we have used queue as data structure to implement our proposed algorithms.

Following is the pictorial representation of our technique:

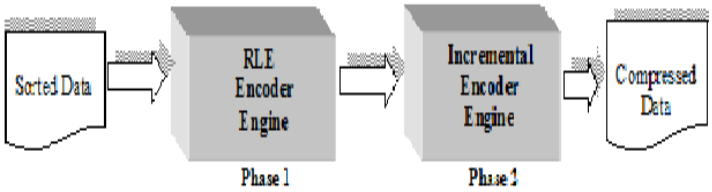

Figure 2: Proposed technique flows

5.2.1 Algorithm for RLE Encoding Engine (Phase 1)

Step 1: Initialize queue; front <- rear <- 0
C.symbol <- P.symbol <- First symbol

Step 2: **if (C.symbol == P.symbol)**
then enqueue C.symbol
rear <- rear + 1
P.symbol <- C.symbol
C.symbol <- Read Next symbol
Repeat Step 2.
else Go to Step 3.

Step 3: **if (rear > 1)**
then print " (rear + 1) P.symbol "
else print " Dequeue all symbols "

Step 4: front <- rear <- 0;
P.symbol <- C.symbol

Step 5: **if (C.symbol == EOF)**
then Exit.
else Go to Step 2.

5.2.2 Algorithm for Incremental Encoding Engine (Phase 2)

Step 1: Initialize Queues (Q1, Q2, Q3)
T.count <- count1 <- count2 <- 0
alphabet1 <- alphabet2 <- NIL

Step 2: Q1 <- NULL;
Q2 <- Q3 <- First Word

Step 3: **if (Q1 == EMPTY)**
then print " Dequeue all element of Q2 "
Go to Step 5.
else Dequeue one element of Q1 and Q2
for i = 1, 2
if (Qi(E) == Alphabet)
then **if (counti == 0)**
then counti <- 1
alphabeti <- Qi(E)
else counti <- Qi(E)
Repeat Step 3.

Step 4: **if (alphabet1 == alphabet2)**
then T.count <- T.count + count2
if (count1 == count2)
then **Go to Step 3.**
else print " T.count '.' Dequeue all elements of Q2 "
else Go to Step 5.

Step 5: Q1 <- Q3; Q2 <- Q3 <- Next Word
T.count <- count1 <- count2 <- 0
alphabet1 <- alphabet2 <- NULL

Step 6: **if (Q2 == EMPTY)**
then EXIT.
else Go to Step 3.

6 Key Features

The proposed technique has the following key features:

- It is Lossless compression technique.
- Compression ratio is very high among lossless compressions.
- Increase the transmission rate because of high compression.
- Increase the disk storage capacity.
- It is a two phase enhanced encoding.

7 Conclusion

The proposed technique gives an excellent result of data compression among Lossless compressions. In this technique we can reduce the size of sorted data by 50% using two phase encoding technique. Comparison table illustrates the compression ratio of RLE algorithm, Incremental algorithm and our proposed technique which shows that compression ratio of our proposed technique is best among of three which is 50%.

References

1) http://en.wikibooks.org/wiki/Data_Compression
2) http://en.wikipedia.org/wiki/Data_compression
3) http://www.ics.uci.edu/~dan/pubs/DataCompression.html
4) http://books.google.co.in/books?id=ChSOjgiY84YC&printsec=frontcover&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false
5) http://datacompression.info/
6) http://www.data-compression.com/index.shtml
7) http://www.cs.cmu.edu/~guyb/realworld/compression.pdf
8) http://www.webopedia.com/TERM/D/data_compression.html
9) Introduction To Data Compression by Khalid sayood(http://books.google.co.in/books?id=ChSOjgiY84YC&printsec=frontcover&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false)
10) http://www.amazon.com/Data-Compression-Book-Mark-Nelson/dp/1558514341#reader_1558514341

11) http://www.cs.cmu.edu/~guyb/realworld/compress.html
12) http://en.wikipedia.org/wiki/Run-length_encoding
13) http://en.wikipedia.org/wiki/Run_Length_Limited
14) http://www.fileformat.info/mirror/egff/ch09_03.htm
15) http://www.data-compression.info/Algorithms/RLE/index.htm
16) http://www.arturocampos.com/ac_rle.html
17) http://www.binaryessence.com/dct/en000045.htm
18) http://michael.dipperstein.com/rle/index.html
19) http://pippin.gimp.org/image_processing/chap_dir.html
20) http://wiki.multimedia.cx/index.php?title=Run_Length_Encoding
21) http://www.prepressure.com/library/compression_algorithms/rle
22) http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?isnumber=6006569&arnumber=6006588
23) http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1055714
24) http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5945475
25) http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5685451