Chapter 1
Introduction to data compression

1. Introduction

Data Compression is a technique of representing information in a compact form. The creation of these compact representations is by identifying and using structures that exist in data. Data can be characters in a text file, numbers that are samples of speech or images. Data compression attempts to identify redundancy in data and use it effectively for compression. This increases the effective density of transmitted or stored data. Data compression consists of two algorithms. First is compression algorithm that takes an input X and generates a representation X_c (compressed data) that requires fewer bits, and Second is a reconstruction algorithm that operates on the compressed representation X_c to generate the reconstruction Y (reconstructed data). Based on requirements of reconstruction, data compression schemes can be divided into two broad categories:

1. Lossless compression schemes: In this scheme, Y (reconstructed information) is exactly same as X (original information). Lossless compression is generally used for applications that cannot tolerate any difference between the original and reconstructed data. Text compression is an important example of lossless compression.

2. Lossy compression schemes: In this scheme, Y (reconstructed information) is not exactly as X (original information), but it is almost same as X. It involves some loss of irrelevant information. Data that have been compressed using lossy techniques generally obtain much higher compression ratios than is possible with lossless compression. Speech, Video and image compression are the important examples of lossy compression.

A very logical way of measuring how well a compression algorithm compresses a given set of data is to look at the ratio of the number of bits required to represent the data before compression, to the number of bits required to
represent the data after compression. This ratio is called compression ratio. The development of data compression algorithms for a variety of data can be divided into two phases [1]: Modeling and Coding:

1.1 Compression Modeling

Modeling refers to extraction of information regarding the redundancy of data and to present this redundancy in the form of model [2]. Models can be static, semi adaptive, or adaptive. Static model uses the same model for all texts. Semi-adaptive modeling employs different models for each input file. It can be constructed during a preprocessing the input and is transmitted to the decoder. Adaptive (or dynamic) modeling does not transmit a model explicitly. Rather, the adaptive model evolves and adapts to changes in the input as data is being encoded. After initializing the model, the encoder compresses one or more input characters based on the current model and subsequently updates the model. The decoder receives the input, decodes it and updates the model in the same way. Lossless compression is implemented by using one of the two different types [3] of modeling:

1. Statistical modeling
2. Dictionary based modeling

1.1.1 Statistical Modeling

It uses a static table of probability. In this we extract information about redundancy that exists in the data and describe the redundancy in the form of a model. This model generally encodes single symbols at a time, reads it, calculate the probability, and then output a single code.
1.1.2 Dictionary Based Modeling

In many applications, the output of the source consists of recurring patterns. A good example is a text source in which certain patterns or words recur constantly. A reasonable approach to encode such sources is to keep a list, or dictionary, of frequently occurring patterns. When these patterns appear in the source output, they are encoded with a reference to the dictionary. If a pattern does not appear in the dictionary then it can be encoded using some other, less efficient method. The efficiency of this method depends upon the class of frequently occurring patterns. The utility of the scheme depends upon the percentage of the words encountered in the dictionary. The average number of bits per pattern can be calculated using the probability of the pattern. Depending on the knowledge, stored in dictionary, we have two approaches:

a) Static approach
b) Adaptive approach

Static Dictionary approach:

This approach is efficient if considerable prior knowledge about the source is available. Mostly used in application like compressing student records, as we know the structure of the complete record. The possibility of recurring patterns in this case is more.

A static dictionary technique that is less specific to a single application is digram coding, where dictionary consist of the source alphabet followed by as many pairs of letters called digrams as can be accommodated by the dictionary. The digram coding reads a two-character input and searches the dictionary to see if this input exists in the dictionary. If it does, the corresponding index is encoded and transmitted. If it does not, the first character of the pair is encoded and second character in the pair then becomes the first character of the next digram. The search procedure is repeated.
Adaptive approach:

The two landmark papers [4] by Jacob Ziv and A. Lempel in 1977 and 1978 provides two different approaches to adaptively build dictionaries, and each approach gives rise to a number of variations. The approaches based on the 1977 paper are said to belong to the LZ77 family are also known as (LZ1 ) and the approaches that are based on the 1978 paper are said to belong to the LZ78, or LZ2 family. To summaries dictionary approach, encoder reads input and look for groups of symbols that appear in a dictionary. If string match is found, a pointer or index in to the dictionary can be outputted instead of code for the symbol. Longer the match, compression ratio will be better.

1.2 Compression Models

There are three ways of building model related to compression:

Physical model:

If we know something about the physics of the data generation process, we can use that information to construct a model. For example Speech related application requires knowledge about the physics of speech process.

Probability Models:

In this by knowing the probability of occurrence to each letter in the alphabet we can compute the entropy of the source.

Morkov model:

One of the most popular ways of representing dependence in the data is through the use of Morkov models, named after A.A. Morkov. For models used in lossless compression a specific type Morkov chain is used. [1]
1.3 Compression coding

Coding refers to description of models and assignment of binary sequences to elements of alphabets. The set of binary sequences is called a code, and an individual member of the set is called codeword. A statistical coder assigns a code to each string based on the probabilities given by the model. For a static or semi adaptive model, Huffman coding, Shannon- Fano coding and arithmetic codes attempts to assign short codes to frequently occur input strings. Codeword- based statistical coders replace input strings by codeword, to obtain a more compact representation of the input.

1.4 Evaluation of Entropy

One of the important issues for given information, is to discover mathematical laws governing systems designed to communicate or manipulate information. It sets up quantitative measures of information and the capacity of various systems to transmit, store and process. Information is related to probability. A high probability event conveys a less information than low probability event.

Information theory uses the term entropy, as a measure of how much information is encoded in message. The higher the entropy of message the more information it contains. The information content $I(x)$ of message (in bits), is defined as:

$$I = - \log_2 P(x).$$  \hspace{1cm} (1.1)

Where, $P(x)$ is probability of occurring an event $x$.

The entropy of an entire message is simply the sum of all entropy of all individual symbols. Entropy fits with data compression in its determination of how many bits of information are actually present in a message. In order to compress data well, we need to select models that predict symbols with high probabilities. A symbol with high probability has low information content and will need fewer bits to encode. Information can be measured by:
Introduction to data compression

\[ H(S) = \sum P_i \log(1/P_i) \]  
\[ (1.2) \]

where \( H(S) \) is the expected information rate for the source \( S \) with the probabilities \( P_i \).

The expected total information, \( \langle I_s \rangle \) for a string of \( n \) independent symbols issuing from \( S \) is then:

\[ \langle I_s \rangle = H(S) \times n. \]  
\[ (1.3) \]

where \( I_s \) is Information string and \( n \) is independent symbols.

The Shannon's entropy [5, 6] states that the average number of selections per symbol is needed to identify the message symbol amongst the \( k \) in the code, but replaced the 'symbols' by 'steps'. Thus the information conveyed by a symbol is, least number of steps needed to identify it amongst the \( k \) in the code. It is very difficult to find whether the minimum number of steps has been achieved for a given string or not. Computational difficulty of deducing the minimum number of steps increases exponentially with the length of the string. This limitation made RPLC (Recursive Linear Pattern Copying) algorithm indicative as a relative measure of information content for individual strings. Mark R. Titchener [7] came up with the review of recursive hierarchical pattern copying (RHPC) which is used as a measure of the effort required constructing a given string. The RHPC complexity of a string \( x \) is then:

\[ CT(x) = \sum \log_2 (k_i + 1) \]  
\[ (1.4) \]

where \( CT(x) \) is complexity of \( x \) and \( k \) is steps.

After comparing the RHPC with other parsing algorithm it is observed that the vocabulary is significantly reduced.
The lower bound for the RHPC complexity can be derived, by considering a string comprising a single repeated character:
\[ CT(x) \geq \log_2 (|x|) \quad (1.5) \]

Figure 1.1 (a) compares the various bounds, including the LZ upper bound, from the above equation, for the RLPC complexity measure, the upper and lower bounds for the RHPC algorithm and the lower bound for the RLPC parser.

The plotted curve using the RLPC and RHPC parser corresponds to random strings generated from the decay events of a radioactive source, and "complex strings" formed by listing sequentially and consecutively all bit patterns of length \( j = 1, 2, 3 \ldots \)

The graph illustrates the significant reduction in the RHPC vocabulary size compared to the RLPC algorithm.

Figure 1.1 (b) compares the RLPC and RHPC complexities respectively for a series of text files. While the forms of the curves is again similar for both algorithms, it is evident that the RHPC algorithm again achieves more compact vocabularies that the RLPC parser, which suggests that significant compression gains may be possible from using an RHPC parser. RHPC is more sensitive to the variation between the sources in spite of the reduced vocabulary sizes.

Figure 1.1: (a) Comparison of bounds for the RLPC and RHPC algorithms and complexity functions for random and complex strings.

(b) Comparison of complexity functions for a series of printed texts.
Figure 1.2, illustrates the computation of complexities, information and entropy functions for a binary scanned documented image. The file is parsed to give a vocabulary related patterns, and associated pattern repetition parameters. From the latter the complexity function is calculated. And by the taking the inverse logarithmic integral results in the information function. Finally the entropy function is computed by differentiating the information function with respect to string length.

Figure 1.2: The computation of complexity, information and entropy functions for an example file: a binary scanned image.
The file is parsed to give a vocabulary of recursively related patterns, and associated pattern repetition parameters. From the pattern repetition parameters the complexity function is computed. The entropy function is computed by differentiating the information function with respect to string length.

To determine the entropy orders of an input file, an algorithm is proposed by Piergiorgio Lanza [8]. This algorithm helps to analyze properties related to entropy orders. The results convey that a random file is incompressible. It has been reported that all pseudorandom files reach exactly zero with in the fifth order. Beside $H_{\text{char}}$ where $H$ is the last character of an input file. He has introduced a parameter $NN$. This NN parameter is number of possible sequence of character. The parameter determines the entropy order specified. It is possible to get following information based on the analysis:

- It is possible to compress the file
- If the file is text file or similar an approximate value of compression ratio is given
- If the file is hard to compress it is possible to adopt a Huffman or first order arithmetic coding to avoid waste of computational resources and time.
- If the file is easily compressible it is possible to decide if it is the case to spend more computational resources to squeeze the file better.

A general technique for reducing entropy coding is symbol grouping [9] where a source alphabet is partitioned, before coding, into small number of groups. Again each data symbol is coded with group number and symbol index. As it uses binary representation to represent symbol index it is faster than any form of entropy coding. The analysis of the complexity of computing the optimal alphabet partition, shows that direct implementation of the dynamic programming recursion needs $O(\text{NS}^3)$ where, $\text{NS}$ is (Number of groups of symbols), operations and $O(\text{NS}^2)$ memory to find all the optimal partitions of an alphabet with $\text{Ns}$ data symbols. The optimal results can yield with a small number of groups. Symbol grouping can be used for context based entropy coding.
The basic lossless data compression Techniques that are considered are:
- Lempel-Ziv Technique
- Huffman coding
- Run length coding
- Arithmetic coding

1.5 Lossy data compression

We are dealing with lossy compression with reference to images. The loss of irrelevant data creates distortion in image. In return for accepting this distortion in the reconstruction, we can generally obtain much higher compression ratios than is possible with lossless compression. The essential merit for data compression is the "compression ratio", or ratio of the size of a compressed file to the original uncompressed file. Much information can be simply thrown away from images, video data, and audio data, and when uncompressed, such data will still be of acceptable quality. But performance measures are necessary to determine the efficiency of lossy data compression schemes, which can be calculated by rate of distortion and Error metrics [10]. Figure 1.3 shows, that the output of the source is modeled as a random variable $X$. The source coder takes the source output and produces the compressed representation $X_c$. The channel block represents all transformations the compressed representation undergoes before the source is reconstructed. By taking the channel to be the identity mapping, which means $X_c = X_c$. The source decoder takes the compressed representation and produces a reconstruction of the source output for the user.

![Block diagram of a generic compression scheme.](image)

Figure 1.3  Block diagram of a generic compression scheme.
1.5.1 Distortion Criteria

In the best of all possibilities we would like to incur the minimum amount of distortion while compressing to the lowest rate possible. The rate distortion theory [11] is studied to minimize rate and keeping the distortion small. To understand the distortion criteria, we would always use the end user of a particular source, to assess quality and provide the feedback required for the design. To look at the fidelity of a reconstructed sequence is to look at the differences between the original and reconstructed values that is the distortion introduced in the compression process. Two popular measures of distortion are squared error measure and the absolute difference measure. These are called difference distortion measures. If \( \{x_n\} \) is the source output and \( \{y_n\} \) is the reconstructed sequence, then the squared error measure is given by:

\[
d(x,y) = (x - y)^2
\]  

and the absolute difference measure is given by

\[
d(x,y) = |x - y|
\]

In general, it is difficult to examine the difference on a term-by-term basis. So, a number of averages Measures are used to summarize the information in the difference sequence. The most often used average measure is the average of the squared error measure. This is called the mean squared error (MSE). And if we are interested in the size of the error relative to the signal, we can find the ration of average squared value of the source output and the MSE. This is called the signal to noise ratio (SNR):

\[
SNR = \frac{\sigma_x^2}{\sigma_y^2}
\]
Introduction to data compression

Where $\sigma_{2x}$ is the average squared value of the source output, or signal, and $\sigma_{2d}$ is the MSE. The SNR is often measured on a log scale and the units of measurement are decibels (DB):

$$\text{SNR}(\text{dB}) = 10 \log_{10} \frac{\sigma_{2x}^2}{\sigma_{2d}^2}$$  \hspace{1cm} (1.9)

We can also find the size of error relative to the peak value of the signal $x$ peak than with the size of the error relative to the average squared value of the signal. This ratio is called the peak signal to noise ratio (PSNR) and is given by PSNR(dB).

$$\text{PSNR}(\text{dB}) = 10 \log_{10} \frac{\sigma_{\text{peak}}^2}{\sigma_{2d}^2}$$  \hspace{1cm} (1.10)

A lower value for MSE means lesser error, and as seen from the inverse relation between the MSE and PSNR, this translates to a high value of PSNR. Logically, a higher value of PSNR is good because it means that the ratio of Signal to Noise is higher. Here, the 'signal' is the original image, and the 'noise' is the error in reconstruction. So, if you find a compression scheme having a lower MSE (and a high PSNR), you can recognize that it is a better one.

The usual steps involved in compressing an image are [12]

1. Specifying the Rate (bits available) and Distortion (tolerable error) parameters for the target image.
2. Dividing the image data into various classes, based on their importance.
3. Dividing the available bit budget among these classes, such that the distortion is a minimum.
4. Quantize each class separately using the bit allocation information derived in 3.
5. Encode each class separately using an entropy coder and write to the file.

Reconstructing the image from the compressed data is usually a faster process than compression. The steps involved are:
Introduction to data compression

1. Read in the quantized data from the file, using an entropy decoder. (reverse of Step 5).
2. Dequantize the data. (Reverse of step 4).
3. Rebuild the image. (Reverse of step 2).

The lossy compression techniques considered here are:

- Vector quantization
- DCT
- Fractal compression
- Wavelet coding
- JPEG and MPEG

1.6 Parallel and Distributed Systems

With serial computing, on Von Neumann style machines, there is one processor which is executing a series of instructions in order to produce a result; there is a logical flow of control through the program. This is true even of programs, such as operating systems, which give the appearance of performing multiple tasks. At any one time there is only one operation being carried out by the processor. Parallel computing [13] is concerned with producing the same results using multiple processors. The problem which is to be solved is divided up between a numbers of processors. Ideally, if a program is running on \( P \) processors, we would like it to go \( P \) times faster than on one processor. In practice this is extremely difficult to achieve due to overheads such as communication time and purely sequential parts of the code e.g., file accesses.

Dividing the problem in a sensible and efficient manner is critical to achieving good performance on a parallel machine. Generally, want to ensure that each processor is performing a similar amount of work means, the program is load balanced and that the work is being distributed as effectively as possible and each processor to be spending the majority of its time on calculations and as little as possible time on communication.
Introduction to data compression

A widely used classification scheme for parallel architectures is Flynn's taxonomy, which labels hardware by instruction stream and data stream. For example, a conventional serial computer would be classified as Single Instruction Single Data (SISD) as the processor executes one instruction on one piece of data at each step. List of the main types of parallel architectures are.

- Single instruction related SISD and SIMD
- Multiple instruction related MISD and MIMD

Typically SIMD consists of many simple processors, each with a local memory in which it keeps the data which it will work on. Each processor simultaneously performs the same instruction on its local data, progressing through the instructions in lock-step, with the instructions issued to the grid of processors by the controller processor. The processors can communicate with each other in order to perform shifts and other array type operations. SIMD architectures are good for problems where the same operation is performed on a number of different objects, for example image processing which acts on each pixel in turn. However, if the loads on the processors are not balanced performance is poor (because execution is synchronized at each step, with everything waiting for the slowest processor).

MIMD architectures typically consist of a small number of independent processors capable of executing individual instruction streams, possibly with each processor executing a different program. Within this class of architectures a common method of subdividing them is on the relationship between processors and memory. This type of division leads to three main types of MIMD architectures. In the shared memory architecture, there are, typically, a small number of processors, each of which has access to a global memory store via some interconnect or bus.

The processors communicate by one processor writing data into a location in memory and another processor reading the data. With this type of communications the time to access any piece of data is the same, as all communication goes through the bus.
The distributed memory architecture avoids the drawbacks of the shared memory architecture by giving each processor its own memory. A processor can only access the memory which is attached directly to it. If a processor needs data which is contained in the memory of a remote processor, then it must send a message to the remote processor asking it to send it a copy of the data. Clearly, access to local memory can occur much faster than access to data on a remote processor. The popular method is to connect the processors in a "hypercube" arrangement. Another type of distributed memory parallel machine which is often overlooked is clusters of workstations. They offer the opportunity for users without the resources to buy massively parallel machines to gain some of the benefits of parallelism on machines available locally, and perhaps not used constantly. We can take advantage of these architectures and concepts to gain speed in lossless and lossy compression processes.

1.7 Objective and Overview of the work

Data compression is often referred to as coding, where coding is a very general term encompassing any special representation of data which satisfies a given need. Information theory is defined to be the study of efficient coding and its consequences, in the form of speed of transmission and probability of error. Data compression may be viewed as a branch of information theory in which the primary objective is to minimize the amount of data to be transmitted.

In this research, we focus on data compression techniques. We are concentrating on lossless data compression. The advantage of lossless data compression is that the compressed file will decompress to an exact duplicate of the original file, mirroring its quality. The disadvantage is that the compression ratio is not all that high, precisely because no data is lost. After reviewing the existing lossless and lossy techniques we have followed the similar principle gaining the compression by removing the redundancies that exist in data.
1.7.1 Proposed approach in brief

In this work an approach has been suggested for lossless compression technique. This approach works for binary form of text and image. We have tried to remove the inter redundant bits to achieve lossless compression. The binary form input file undergoes the encoding process which results in compressed file which in turn goes to decoding procedure to obtain the original file. The algorithm has been developed using MATLAB and tested for 10 different image files and text files.

Organization of the thesis:

The thesis is organized in VI chapters. Chapter 2 is a reference chapter which attempts to establish the fundamental of data compression and explores the different techniques fall under lossless category of data compression. It explains the basic algorithm of Lempel-Ziv, Hoffmann coding, Run length coding and Arithmetic coding. It is also devoted for the recent advances and variations of above data compression techniques.

Information can be in the form of text or image. Data compression works on image by lossy techniques. Chapter 3 deals with the fundamentals of lossy data compression with the advances done in this field. It also explores the techniques of image compression like DCT, VQ, Fractal compression, Wavelet coding, JPEG and MPEG.

Since the dawn of time, almost all electronic computers have the same general form a single processor connected to a single bank of memory appearing to execute a single program at a time. But in late 1960’s many computer architects came forward to explore alternatives in which many processors worked together on the same problem at the same time. More and more use of computers leads to more use of information. This use of information requires space for storage and time for transmission. Large information needs large space and more time for transmission which is not suited for many applications. The sharing of information is the main motivation for the computer networking. To solve the problem of large storage and transmission time, we look for compression and some mechanism of fast transfer, thus the ultimate solution is the use of parallel and distributed systems. The
combination of data compression in parallel and distributed processing is the need of present computer world. Chapter 4 is devoted for data compression in parallel and distributed systems. It focuses on the research done in this area with the advances. This chapter is devoted to study advanced computer architectures like pipelined computers, array processors and multiprocessor systems.

Motivated to gain compression, new algorithms are proposed in Chapter 5. We have implemented the proposed algorithm on text and image files. The experiment has been carried on ten different text files and image files. The results when compared with the Huffman coding, Lempel Ziv and Run length coding shows efficient compression ratio. The parallel system for the proposed algorithm is also presented. We have also presented an approach for cryptography which found to produce good cipher text.

Based on the results of the experimental work, Chapter 6 includes the conclusion of the research work with the future aspects.
References

   zSz–danzSzpubsSzTR91-44r.pdf/stauffer93parallel.pdf
3. Ranjan bose,“Information, Theory and Coding and Cryptography”, Tata
5. Gagie T ,"Dynamic Shannon Coding”, Page(s) : 540, Digital Object
   Identifier 10.1109/ Data Compression Conference 2004.
6. Tjalling Tijalkens,"Implementation Cost of the Huffman Shannon Fano
   Object Identifier 10.1109/ Data Compression Conference 2000.
8. Lanza P, “Entropy and Data Compression Performance”, Page(s): 558,
   Entropy Coding “, Data Compression Conference, 2005.
10. www. netnam.vn/ unescocourse/ computervision/10.htm
11. R.C. Gonzalez, R.E.Woods, S,L Eddins, "Digital image processing using