PREFACE

I have made this letter longer than usual
because I lack the time to make it shorter.
Blaise Pascal (1623 – 1662)

Research findings have proved that redundancy in data representation is considerably reduced by compression algorithms as they help decrease the storage required for that data. An able and judicious administration of available bandwidth by data compression results in reduction of communication costs. Research in the last ten years has triggered an unprecedented explosion in the volume of digital data transmitted over the Internet, representing text, images, video, sound, computer programs etc. The future holds a much greater prospect for digital data explosion if the present momentum of research is stepped up and moves into top gear; and then, hopefully, much improved algorithms, born of relentless research, will result in a remarkably maximal compression of data through effective use of available network bandwidth.

This unique outbreak of textual information over internet leads to the requisite Data compression as the way out to the challenge. It is, in fact, charming to figure out that data compression and its wide techniques have smoothed the progress of this transformation.

Data compression is a technique that reduces the size of the data by the process of encoding the data in such a way that fewer bits are needed to represent the data than the original. The core objective of data compression is to represent different types of data such as text, image, sound and video that are transmitted over the network in a digital form with as few bits as possible. This is accomplished by removing redundancy in the data which determines the success of compression.
Data compression can be achieved by different means. Not every method is suitable for every type of data. Each method takes up dissimilar thought for dissimilar type of data and achieves compression. But the commonality among them lies in the principle they stick to – removing redundancy in the data. Thus, data compression embraces the identification of redundant data and removal of such redundancy from the data which, in turn, reduces the memory consumption by reducing number of bits used for storing the data. The process is carried out by means of specific encoding schemes. Thus, the original text after applying encoding scheme by a compressor will be converted into a compressed form.

Redundancy in data refers to patterns in the data source, to data being repeated in a data source or to discernible differences in the data source. Terry Welch identifies four different types of redundancy that exist in a data source without explicit knowledge of how the data is interpreted. They are distribution of characters, replication of characters such as spaces, high-usage of certain patterns or sequences and positional redundancy. Redundancies within a source file are, thus, exploited by different compression techniques to create a compressed data source.

Data compression is classified into two major sets such as lossless and lossy data compression based on the possibility of reconstructing the original data. The term lossless compression refers to the aspect of compression in which no loss of data crops up while decompressing. The actual data before compression will be exactly recovered to its original form after it gets decompressed. Lossless compression can be utilized where no loss of data should be tolerated. It is applied when it is important that the original and the decompressed data need to be identical, or when no assumption can be made on whether certain deviation is uncritical.
Lossless data compression is used in many applications. Typical examples are executable programs and source code. Some image file formats, notably PNG, use only lossless compression. In particular, loss of data cannot be compromised in Medical and Space-associated images.

The term lossy compression refers to loss of data while decompressing. This is defined as an irreversible process since the actual data is not regained when decompressed. This type of compression can be carried out in the case where exact data is not imperative. In the case of multimedia files such as images, audio and video where the quality of the file, even when affected, goes undetected, lossy compression can be carried out.

Habitually lossless compression algorithms are carried out in two steps: generation of statistical model for the data in the first step and in the second step, the model is used to map the data to bit sequences which are shorter for frequently encountered data and comparatively longer for less frequent data.

The lossy compression methods can be used advantageously as they yield much better compression ratios than lossless algorithms, if allowances can be made for some distortions of data. In many countries there is a legal ban against the use of lossy methods to compress picture data. Hence medical images are compressed only by the lossless algorithms.

One approach for trying to attain better compression ratios is to develop different compression algorithms. A number of sophisticated algorithms have been proposed for lossless text compression such as Huffman encoding, Arithmetic encoding, Lempel Ziv family, Dynamic Markov Compression (DMC), Prediction by Partial Matching (PPM) and Burrows Wheeler Transform (BWT) based algorithms. However, none of these methods has been able to reach the theoretical best-case compression ratio consistently, which suggests that better algorithms may be possible.
One of the prime factors that leads to developing compression methods is to prepare a specialized compression algorithm for the data to be transmitted or stored. In general, it is not viable to prepare a specialized compression method for each type of data. The reasons are two-fold.

- It would result in a vast number of algorithms.
- The cost of developing a new compression method could be much higher than the gain obtained by the reduction of the data size.

The main handicap in this is nothing about the data can be assumed. The excessive information can not be found out through any means. So, a compromise is required. The defining of the classes of sources producing different types of data forms the standard approach in compression. It is assumed that the data are produced by a source of some class and a compression method applied for that particular class is applied. The algorithms are termed universal if they are found working well on the data that can be approximated as an output of some general source class.

Lossless compression techniques fall under two different categories as statistical-based and dictionary-based techniques. Frequency of input characters happens to be the source of compression in statistical-based techniques. This method assigns code to all the symbols based on the frequency of occurrence. Hence, in this manner the symbols with higher probability acquire shorter codes and those with lesser probability, comparatively the longer codes. Under this category, the two major and very popularly used algorithms for producing the bit sequences are Huffman coding [46] and Arithmetic coding [2].

In Huffman coding fixed length codes are replaced by variable length codes, allocating shorter code words to symbols with higher frequency. This technique experiences the fact that an integral value of bits is needed to code a
character. As an alternative, Arithmetic coding replaces a stream of input symbols with a single floating point number instead of replacing every input symbol with a codeword.

Dictionary-based techniques replace input strings with earlier identical input. The focal concept of these algorithms is to identify repetition pattern from a string. These algorithms do not encode single symbols as variable bit strings. Instead they encode variable length strings of symbols as single tokens in such a way that the tokens form an index to a phrase dictionary.

Today a number of dictionary-based lossless data compression methods are being increasingly used. Historically the first ones were introduced by Ziv and Lempel during 1977–78. It was a pursuit of the authors to look for the data to compress for identical parts and to replace the repetitions with the information where the identical subsequences appeared before. There are many tasks in which this task can be effectively carried out. Ziv and Lempel Ziv and Lempel proposed two main variants of their method: LZ77[99], which encodes the information of repetitions directly, and LZ78[100], which maintains a corresponding dictionary of subsequences appeared so far, and stores the indexes from this dictionary in the output sequence. The following are the two advantages of these methods: high speed and ease of implementation. However, their compression ratio is worse than the ratios obtained by other contemporary methods. LZW, proposed by Welch is an improved scheme, maintains a dictionary that is indexed by codes and is assumed to be initialized with 256 entries representing the ASCII table. In this algorithm, one character at a time is read and if the code exists in the dictionary, it adds the character to the current work string, and stay for the next one. This happens on the first character as well. In case the work string is not in the dictionary, the string is then added to the dictionary and transmits the code assigned to the work string without the new character. It then sets the work string to the new character.
An alternative approach, which is taken up for analysis in this thesis, seeks to perform a lossless, reversible transformation known as preprocessing algorithm to a source file before applying an existing compression algorithm. The compression of the source file in a much easier way is accomplished through the said transformation. The input to the transformation is the source file whereas the transformed text is the output. The output is fed to an existing data compression algorithm and the outcome is an efficient compression of the transformed text. A reversal of this process is required to be done for decompression by first invoking the appropriate decompression algorithm, and then providing the resulting text to the inverse transform. The preservation of the overall lossless text compression paradigm without any compromise is totally dependent upon the exactly reversible transformations. The data compression and decompression algorithms are unmodified, so they do not exploit information about the transformation while compressing. The approach is used with the singular objective of enhancing the compression ratio of the text in comparison with what could have been achieved by using only the compression algorithm. It has been observed that the preprocessing of the text prior to conventional compression will improve the compression efficiency much better.

**Thesis Contribution**

In this thesis, attention is focused on Dictionary-based preprocessing. It is a technique for compressing texts in a natural language. The basis of the technique is replacing whole words with shorter codes. This results in a very considerable improvement in the compression effectiveness among textual preprocessing methods. The dictionary of words is usually fixed for a specific language and has to be prepared in advance. This dictionary is commonly used by both the sender and the receiver.
Some inherent redundancy is inevitable in the natural languages like English, as not every combination of letters represents a proper word. Texts in natural languages have specific structure. They can be divided into sentences, finished by a period, a question mark, or an exclamation mark. Each sentence consists of words that are separated from the neighboring words by space and/or punctuation marks. The main drawback is that the compression algorithms already in use do not completely exploit this redundancy. Thus, there is a need to develop effective preprocessing algorithms which can fully exploit the natural redundancy of the language in a much better way and it has been observed that the preprocessing of the text prior to conventional compression will improve the compression efficiency much better.

The objectives of this research are:

- To carry out a detailed study of compression algorithms.
- To study and analyse the existing dictionary-based preprocessing algorithms.
- To identify the methodologies that improve the effectiveness of the preprocessing algorithms.
- To design and develop new preprocessing algorithms.
- To experiment the algorithms.

In this thesis, four new dictionary-based lossless preprocessing methods are proposed for better compression. The methods are listed below.

- Dictionary-Based Encoding Using Alphabetic Indexing (DBEAI)
- Enhanced Intelligent Dictionary-Based Encoding (EIDBE)
- Improved Intelligent Dictionary-Based Encoding (IIDBE)
- Reinforced Intelligent Dictionary-Based Encoding (RIDBE).
The structure of the thesis is as follows:

Chapter 1 describes the Data compression and its types. In addition to that, applications of lossless and lossy compression techniques are given. It also explains about the methods of coding symbolic data, elementary information theory and the classification of lossless data compression algorithms such as statistical methods and dictionary methods.

In Chapter 2, three different lossless data compression techniques are explained. The first compression technique called Statistical compression and its methods like Run-length encoding, Shannon –Fano coding, Static Huffman coding, Adaptive Huffman coding and Arithmetic coding are described. Next is undertaken a practical comparison of statistical compression techniques and the results are discussed in detail. The second technique Dictionary-based compression and its various methods like LZ77, LZ78, and LZW etc are highlighted. Here, a detailed discussion of the results of the practical comparison of LZ77 and LZ78 families are presented. The third predictive compression technique and its various methods like PPM, PPM*, PPMZ and Fast PPM are also described.

Chapter 3 provides the literature review of text preprocessing techniques for data compression. It includes modern paradigm of text preprocessing and need for preprocessing algorithms. This chapter also discusses the dictionary-based compression algorithms. It also lists out the leading dictionary-based preprocessing algorithms such as BWCA, Star transformation, LIPT, NIT, LIT, StarNT, IDBE, DBT, DRT, WRT and TWRT.
In Chapter 4, the proposed method-I called “Dictionary-based encoding with Alphabetic Indexing (DBEAI)” is described. This chapter initiates related work and follows it up with the proposed model description with experimental results.

Chapter 5 proposes the second method called “Enhanced Intelligent Dictionary-Based Encoding (EIDBE)”. It gives a clear picture of the proposed model which comprises dictionary creation, encoding and decoding phases. This chapter also provides the experimental results of the proposed algorithm.

The third proposed method called “Improved Intelligent Dictionary-Based Encoding (IIDBE)” is modeled and explained in Chapter 6. It is outlined with three phases namely dictionary creation, encoding and decoding. The experimental results and performance measures are also highlighted.

Chapter 7 provides yet another model named as “Reinforced Intelligent Dictionary-Based Encoding (RIDBE)” with three tasks like dictionary creation, encoding and decoding. This chapter also contains the experimental results of the proposed algorithm.

Chapter 8 concludes the research work and suggests interesting possibilities for the further research.