Textual Data Compression using Octovicentimal Strange Number System

Debasis Das **Dr U A Lanjewar**

***Abstract* — While technology keeps growing, the world keeps shrinking. Everything seems to be nearer and smaller to everyone. Our world has changed a lot from an era where a computer used to occupy a room to the present where supercomputers can be conveniently carried in your hand. It would be an understatement to merely term this transformation as a technological growth; rather, it should be termed as a 'Technological Explosion'. The primary objective of data compression algorithms is to reduce the redundancy in data representation in order to decrease data storage requirement. Data compression also offers an attractive approach to reduce the communication cost by effectively utilizing the available bandwidth in the data links. Here, we propose a new approach towards compression of text data based on a cryptic representation of text. One basic assumption of our algorithm is that the system has converted all the texts into strange number system. We advocate a different data compression paradigm in this paper.**

***Key Words* — Strange Number System, Octovicentimal SNS, Data Compression.**

I. Introduction

Data compression is the most considerable part in recent world. People have to compress a huge amount of data so as to carry from one place to other or in a storage format. Though a lot of research and findings are already happened in the field of data compression but there are no such algorithms in data compression that lay emphasis on differential compression based on strange number system. The general principle of data compression algorithms on text files is to transform string of characters into a new string which contains the same information but with new length as small as possible. Data compression using strange number system is a novel concept and the potential advantage of these algorithms is their simplicity. An entirely different technique is employed to reduce the size of text files.

Data compression is a technique used on computer systems to compress data, such that the data occupies less storage space, or can be transmitted in shorter time. Data Compression is a division of applied mathematics concerned with developing schemes and formula to reduce the amount of storage or transmission capacity through the use of codes. On computer systems, all information or data is stored as sequences of zeros and ones, regardless of whether the data represents an English text, a complete movie, or anything in between. On such systems, a data compression program tries to reduce the size of information using lesser number of bits than an un-encoded representation would use to reduce the amount of space or time needed to transmit data. Nowadays, data compression is a common requirement in telecommunication industry. Since last few decades, the research is continuing in the field of compression and various data compression techniques are already developed to compress different data formats [1].

This thesis describes the design and implementation of a text compression technique. Text compression – the term should not be taken too literally; – is small but a very lively corner of the data compression world. It is the field of universal lossless data compression techniques used to compress a type of data typically found on computer systems.

II. Text Compression

Text compression is the process of encoding text information using fewer bits. It is the process of restoring compressed data back into a form in which it is again useful. Fig.1. shows the flow chart of text compression and decompression process.

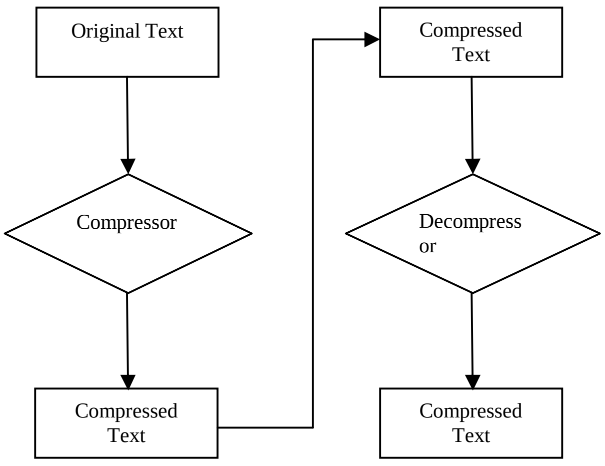


Fig.1. Flow Chart of Text Compression and Decompression Process

The Text Compression has played a pivotal role in achieving text information. Basically there are two types of compression techniques viz. the lossless data compression techniques and the lossy data compression technique. The lossless compression technique is basically used for text compression and the lossy compression technique is typically applied to image data [2]. The original text has lot of redundant bits and compression removes this redundancy by the hardware called as a compressor. Thus the storage size required for the compressed text is reduced. At the end of decompression, the original text is retrieved back as shown in Fig. 8.1.

III. Data Compression: From Ancient Age to Computer Age

Data compression is the science and art of representing the information in a compact form rather than its original or uncompressed form for data storage or transmission. It can be viewed as the art of creating shorthand representations for the data even today, but this started as early as 1,000 BC. From that time a variety of procedures are available specially designed to fulfill different requirements of data compression. Selecting a proper compression procedure is always a compromise that tries to reconcile contrary characteristics. The Braille code was developed for intuitive data compression by Louis Braille in 1829. But the beginnings of actual data compression began as early as 1838 with its use is Morse Code for telegraphy which was based on using shorter code words for letters such as 'e' and 't' that are more common in English. Over 100 years later, as the computer age was on the rise, this simple Morse code method was built upon and became a study that is known as Information Theory. However, even before Information Theory, Fourier's method was applied in data compression; even this methodology is still used today in compression algorithms. So, 1930 onwards analog compression method is used in the field of data compression. Fourier's method was also applied to images in the 1950's in an attempt to decrease the amount of data needed to be sent on television; however, no solution was given at that time. In fact, it took another 30 years until the notion of image compression became prevalent in mainstream technology.

But, modern work on data compression began in the late 1940s with the development of Information Theory. In 1949, Claude Shannon and Robert Fano devised a systematic way to assign code words based on probabilities of blocks. In 1951, David Huffmann found an optimal method for data compression. Early implementations were typically done in hardware, with specific choices of code words being made as compromises between compression and error correction.

The area of Wavelet Compression was founded by Alfred Haar, a Hungarian Mathematician, in the early 20th Century [8]. Haar created the first known Wavelet, which is now known as a Haar Wavelet. This area of compression has grown rapidly since the 1970's when interest in Wavelets and their uses began to spread through the mathematical field.

In the mid 1970s, the idea emerged of dynamically updating code words for Huffman's encoding based on the actual data encountered. And in the late 1970s, huffman coding has been replaced by arithmetic coding by Rissanen. It is superior in most respects to the better-known Huffman's method.

And in the late 1970s, when online storage of text files became common, software compression programs began to be developed, almost all based on adaptive Huffman's coding. In 1977 Abraham Lempel and Jacob Ziv suggested the basic idea of pointer-based encoding. In modern data compression, there are two main classes of dictionary-based schemes named after Jakob Ziv and Abraham Lempel, who first proposed them in 1977 and 1978 [9, 10]. These are called LZ77 and LZ78, respectively.

A few years later, in 1984, Cleary and Witten introduced a prediction by partial matching (PPM) algorithm [11]. In 1987, Michael Barnsley was at the top of field in the development of fractal compression, and currently has many patents on these algorithms. Another statistical compression method, a Dynamic Markov Coder (DMC), was invented by Cormack and Horspool [12, 13] in 1987.

In the mid-1980s, work by Terry Welch, the so-called LZW algorithm rapidly became the method of choice for most general-purpose compression systems. It was used in programs such as PKZIP, as well as in hardware devices such as modems. In the late 1980s, digital images became more common, and standards for compressing them emerged. In the early 1990s, lossy compression methods (to be discussed in the next section) also began to be widely used.

Most of you will be familiar with the term 'Mp3' associated with audio files. Basically, 'Mp3' is a standard for audio compression. 'Mp3' is a part of Motion Picture Experts Group (MPEG). In January 1988, MPEG was established as a subcommittee of International Standards Organization (ISO)/International Electro-technical Organization. It might be interesting to know how the 'Mp3' standard for audio files has evolved. Karlheinz Brandenburg is often known as father of 'Mp3'. In April 1989, Fraunhofer received a German patent for 'Mp3'. In 1992, Fraunhofer and Dieter Seitzer's audio coding algorithm was implemented in MPEG-1. MPEG-1 was published in 1993. In 1994, MPEG-2 was developed and published a year later. US patent for 'Mp3' was issued on 26th November, 1996.

In 1995, an interesting compression method, a Context Tree Weighting (CTW) algorithm, was proposed by Willems et al [14].

Another compression method proposed recently is a block-sorting compression algorithm, called usually a Burrows–Wheeler compression algorithm (BWCA). The authors invented this method in 1994. Without Fractal compression method, streaming information over the internet would have taken many times longer. Although Compressive Sensing has been studied for over 40 years, it did not see significant strides until 2004. At that time, a mathematician, by the name of Emmanuel J. Candes, was performing research with magnetic resonance imaging. He found that an image could be reconstructed even when the data seemed insufficient by the Nyquist-Shannon criterion.

Current image compression standards include: FAX CCITT 3 (run-length encoding, with code words determined by Huffman coding from a definite distribution of run lengths); GIF (LZW); JPEG (lossy discrete cosine transform, then Huffman or arithmetic coding); BMP (run-length encoding, etc.); TIFF (FAX, JPEG, GIF, etc.). Typical compression ratios currently achieved for text are around 3:1, for line diagrams and text images around 3:1, and for photographic images around 2:1 lossless, and 20:1 lossy [15].

Now even the technology master IBM has entered into data compression research by acquiring the data compression company Storwize. In addition, IBM is poised to play a major role in the field of real-time data compression. The future of data compression is very promising as the dependency on compression is getting increased. For example, the WebTV, HDTV, and many gaming applications may get benefitted in future. Also more advanced modems with efficient compression algorithms can lead to smarter exchange of data through web. In addition, NASA, Canadian, European, and Japanese space agencies are together planning for a program to monitor the global change that generates 0.5 TB data per day. So it is crystal clear that it is surely going to get the benefits of data compression. We shall hope for many new innovations in the field of data compression in future which will surely take technology to greater heights.

Finally, data compression techniques have to be universal. This means that they should be able to process all types of data found on computer systems. It is impossible for a data compression program to compress all possible computer files, but at least it should be able to compress a wide range of different types of files.

IV. Strange Number System & Octovicentimal Number System

Number system is used just around everywhere. People count by tens and machines count by twos-that pretty much sums up the way we do arithmetic on this planet. The cultural preference for base 10 and the engineering advantages of base 2 have nothing to do with any intrinsic properties of the decimal and binary numbering systems. But there are countless other ways to count. All the number systems other than the traditional number systems (viz. decimal (base 10), binary (base-2), octal (base-8) and hexadecimal (base 16)) are coined under the strange number systems. There can be various examples of strange numbers with their respective benefits. For example ternary number system (base 3) which uses only three symbols 0, 1, 2 is said to be more convenient than binary and is the root for the emerging field of quantum computing whereas quaternary number system (base 4) which uses only four symbols 0, 1, 2, 3 can be used to perform carry free arithmetic operations. So, apart from the traditional number systems, the strange number system also plays a significant role in computing. Strange number systems are not as widely known or widely used as traditional number systems in computing, but they have charms all their own having a genuine mathematical distinction in its favor. By one plausible measure, it is the most efficient of all integer bases; it offers the most economical way of representing numbers. Today, the complexity of traditional number system is steadily increasing in computing. Due to this fact, strange number system is investigated for efficiently describing and implementing in digital systems. In computing the study of strange number system will be useful to all researchers and knowledge seekers. Their awareness and detailed explanation is necessary for understanding various digital aspects.

Inception digital devices have been designed using binary number system till date. Researchers found that the development in binary number system is cumbersome, complex and difficult to understand. Strange number system enables more information to be packed in a single digit; hence, researchers have been working on strange number system since many years [3].

Strange number system offers important advantages like more information can be processed over given set of lines to reduce the burden of interconnections & there by switching [4]. The information content per interconnection can be raised from the present binary number system to strange number system.

A. Octovicentimal Number System

The number system with base one hundred twenty eight is known as the octovicentimal number system. In this system one hundred twenty eight symbols are used to represent numbers and these symbols are described in Appendix C. It is also a positional number system that each bit position corresponds to a power of 128. It has two parts, the integral part or integers and the fractional part or fractions, set apart by radix point. For example (4z8.l3)128

In octovicentimal number system the leftmost bit is known as most significant bit (MSB) and the right most bit is known as least significant bit (LSB). The following expression shows the position and the powers of the base 128:

…..1283128212811280.128-1128-2128-3…..

The arithmetic operations like addition, subtraction, multiplication and division operations of decimal numbers can be also performed on octovicentimal numbers.

V. Data Compression using Octovicentimal Strange Number System

A. Algorithm Strategy

First we need to read all the words in the input file and count the occurrence of each word. Then sort the words according to the order of their occurrence in descending order and store them in a file. If any word appears more than twice, replace the same using a special character and then modify the original file by replacing those words with their respective assigned special characters. Now we need to count the number of different characters from the modified file and store them in a sequential file. Now, separate four characters at a time from the distinct character set and store them in an n×4 matrix, where n indicates the number of rows and 4 indicates the number of columns of the matrix. Here, number of column is fixed and the column index will be 0, 1, 2 & 3. Now concatenate the entire column index and convert it into octovicentimal number system. But, here, the number of row index depends upon the total number of distinct characters and convert each row index into octovicentimal number system. Then write converted column index after each converted row index. In this way the whole file can be compressed and in the reverse way it can be decompressed.

B. Algorithm

Steps for Compression

1. Start
2. Read all the words in the input file.
3. Initialize a counter for each word.
4. Sort the words according to the number of their occurrences in descending order.
5. If a word appears more than twice, replace it with a special character (ASCII range 128-254) and maintain a dictionary or replaced words in an array Special [].
6. If the count of special characters used reaches 127, combine

two special characters to replace the further words.
7. Now again read all the strings from this modified input file.
8. Store the distinct characters in a jagged array Distinct [ ].
9. Find out the number of distinct characters in this modified file.
10. Separate four characters at a time from the distinct character set and store them into a two dimensional array Final [] [4].
11. Do until end of file.
12. Concatenate four column indexes and convert it into octovicentimal number system.
13. Convert each row index into octovicentimal number system.
14. Write converted column index after each converted row index.
15. Put these converted data into a new Result file.
16. Repeat the process from step 9.
17. End the process.
18. Result file is the final compressed file.
19. End

Steps for Decompression
1. Begin
2. Do until end of compressed string or File.
3. Takes two bits at a time and then calculate their decimal value.
4. Here, first decimal value is the row index.
5. Separate each digit from the second decimal value and these are the column index.
6. Fetch all the elements from each array index and store them in an array Rem[].
7. Now map the each element of the array Rem[ ] with the array Distinct[ ] which is mentioned in Compression routine.
8. Let Rem[ ]={c0, c1, c2, …..,cg-2, cg-1}.Then put the Distinct[c0], Distinct [c1],…. Distinct [cg-1] into a file Extract.
9. Go to step 2
10. End of Loop
11. Now map each special character appearing in the file Extract from the dictionary Special [ ] mentioned in the Compression routine and replace them to retrieve the final Decompressed file.
12. Store the final results into a file Get.
13. End

C. Algorithm Illustration

Consider a text file written:

"My name is Rahul. Rahul is a good boy. Rahul lives in Kolkata."

Here 'is' and 'Rahul' appears twice so these words will be replaced by some special characters say '$' and '#'.

Then the modified file will become like this:

"My name $ #. # $ a good boy. # lives in Kolkata."

Now, the count of total number of characters in this modified file is 45 but the distinct character set is [M, y, n, a, m, e, $, #, ., g, o, d, b, l, v, s, K, k, t, ,]

Then separate four characters from the distinct character set and store them into a matrix where total number of row will be five and the number of column is fixed for each row index.

So, after converting these row and column index into octovicentimal number system the modified text will be- 0ϕ1ϕ2ϕ3ϕ.

So, by representing in this way, the total size of the compressed file will be 56 bits whereas the original size of the text file was (60*8) bits=480 bits.

Hence, a compression percentage of around 88.34% is achieved.

This way, the whole file can be compressed thereby achieving a better compression ratio.

VI. Data Compression Tool using Strange Number System

This software is about building a text compression and decompression software using strange number system. To perform text compression and decompression operation we used a new algorithm using strange number system i.e. Compression using Octovicentimal SNS. This software is implemented in C#.Net platform. Main aim of this software is to develop an application which can convert given information in to compressed format using different algorithms using strange number system and it also converts compressed data to Original format (Decompression) using the same.

A. Main Features

- Text Compression

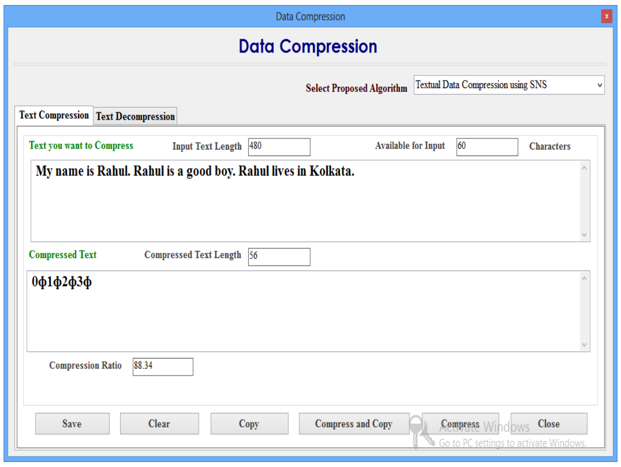
Compression provides secure and reliable way to reduce the size of data and store in a compact manner. The text compression module allows you to compress any texts, documents etc. in seconds.

- Safely Store Information on your Computer using less memory

Compression provides secure and reliable way to store your data in less memory space. Just select what you want to compress, and this software helps you keep documents, private information and files in a concise way and saves your memory.

- Strong Compression

This software does compression using Octovicentimal SNS compression algorithm.


Data Compression

Data Compression

Select Proposed Algorithm: Textual Data Compression using SNS

Text Compression | Text Decompression

Text you want to Compress — Input Text Length: 480 — Available for Input: 60 Characters

My name is Rahul. Rahul is a good boy. Rahul lives in Kolkata.

Compressed Text — Compressed Text Length: 56

0ϕ1ϕ2ϕ3ϕ

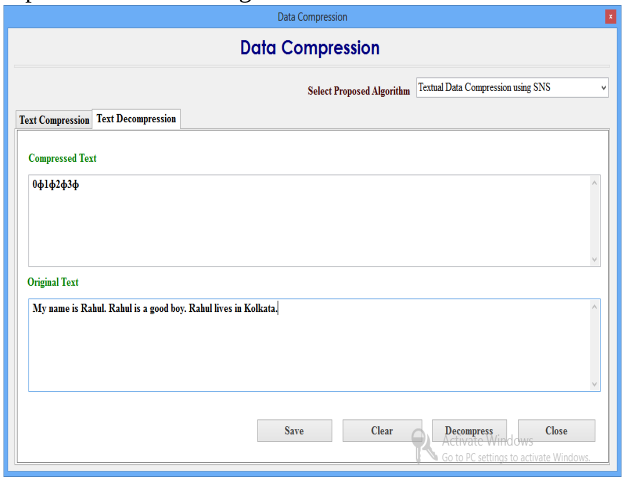
Compression Ratio: 88.34

Save | Clear | Copy | Compress and Copy | Compress | Close

Activate Windows
Go to PC settings to activate Windows.

Fig.2. Text Compression

The system will allow user to enter a message, select a compression method, and then view the message compressed by the selected algorithm. When a compression method is selected, options pertaining to that specific algorithm will be displayed for the user to customize. When the user presses the "Compress" button, the algorithm will then begin compressing the text. After each calculation, text will be displayed to the user related to the operation that has just been performed. It also displays the compression ratio of the given text.


Data Compression

Data Compression

Select Proposed Algorithm: Textual Data Compression using SNS

Text Compression | Text Decompression

Compressed Text

0ϕ1ϕ2ϕ3ϕ

Original Text

My name is Rahul. Rahul is a good boy. Rahul lives in Kolkata.

Save | Clear | Decompress | Close

Activate Windows
Go to PC settings to activate Windows.

Fig.3. Text Decompression

The system allows user to decompress the text message which was already compressed using some compression technique. When the user presses the "Decompress" button, the algorithm will then begin decompressing the encrypted text. After calculations original text message will be displayed to the user which was previously entered for compression.

CONCLUSION

In this paper a new algorithm is used for data compression using octovicentimal strange number system. The potential advantage of this algorithm is its simplicity. It is a simple compression and decompression process, is used to reduce the size of text files. The technique of 'reduce bits' is used in this algorithm. Since every character is taken care of, so the output codes do not depend upon the repetition, like most of the other compression algorithms. Data compression using strange number system is reduced to reduce a larger combination of characters by a fewer numbers of bits. After the code formation, ASCII replaces the octovicentimal numbers, which finally reduces the text size. Though a lot of research and findings are already happened in the field of data compression but there are no such algorithms in data compression that lay emphasis on differential compression based on strange number system and bit reduction.

ACKNOWLEDGMENT

I would like to thanks Professor Ujwal A. Lanjewar sir for his valuable advice, support and constant encouragement. His constant criticisms and reviews gave me the conceptual clarity. I would also like to thank him for being an open person to ideas and for encouraging and helping me to shape my interest and ideas.

REFERENCES

[1] Ziv, Jacob, and A. Lempel. 1977. A Universal Algorithm for Sequential Data Compression. IEEE Transactions on Information Theory. IT-23(3):337–343.

[2] Ziv, Jacob and A. Lempel. 1978. Compression of Individual Sequences via Variable-Rate Coding. IEEE Transactions on Information Theory. IT-24(5):530–536.

[3] Debasis Das, Dr. U A Lanjewar, "Realistic Approach of Strange Number System from Unary to Decimal," International Journal of Computer Technology and Applications, vol. 3(1), pp. 235–241, January 2012.

[4] Debasis Das, Dr U A Lanjewar, ‘’ Exploring Strange Number System: Latent Talent to be used in place of Traditional Number System", International Journal of Advances in Science and Technology, Vol. 3, No.1, 2012.

[5] Yokoo, Hidetoshi. 1991. An Improvement of Dynamic Huffman Coding with a Simple Repetition Finder. IEEE Transactions on Communications. 39(1):8–10. January.

[6] Wu, Xiaolin. 1996. An Algorithmic Study on Lossless Image Compression in James Storer ed. DCC '96. Data Compression Conference. Los Alamitos. CA. IEEE Computer Society Press.

[7] Witten, Ian H., T. C. Bell, H. Emberson, S. Inglis, and A. Moffat. 1994. Textual image compression: two-stage lossy/lossless encoding of textual images. Proceedings of the IEEE. 82(6):878–888. June.

[8] Debashish Chakraborty, Sandipan Bera, Anil Kumar Gupta, Soujit Mondal, "Simple Data Compression by Differential Analysis using Bit Reduction and Number System Theory", ACEEE Int. J. on Information Technology, Vol. 01, No. 03, Dec 2011.

[9] Witten, Ian H., Radford M. Neal, and John G. Cleary. 1987. Arithmetic Coding for Data Compression. Communications of the ACM. 30(6):520–540.

[10] Debasis Das, Dr U A Lanjewar, "Design an Algorithm for Data Compression using Pentaoctagesimal SNS", International Journal of Computer Applications, Volume 62, No.14, January 2013.

[11] Williams, Ross N. 1991a. Adaptive Data Compression. Boston. MA. Kluwer Academic Publishers.

[12] Williams, Ross N. 1991b. An Extremely Fast Ziv-Lempel Data Compression Algorithm. in Proceedings of the 1991 Data Compression Conference. J. Storer, ed., Los Alamitos. CA. IEEE Computer Society Press. pp. 362–371.

[13] Y. M. Kamir, M. Deris. M. Sufian, and A. A.F. Amri. 2009. Study of Efficiency and Capability LZW++Technique in Data Compression. World Academy of Science. Engineering and Technology 35 2009.

[14] P.G.Howard and J.C.Vitter, Fellow IEEE. Arithmetic Coding For Data Compression.

[15] Debasis Das, U A Lanjewar, and S. J. Sharma, "The Art of Cryptology: From Ancient Number System to Strange Number System", International Journal of Application or Innovation in Engineering & Management (IJAIEM), Volume 2, Issue 4, April 2013.

[16] S. Kaur and V.S.Verma. 2012. Design and Implementation of LZW Data Compression Algorithm. International Journal of Information Sciences and Techniques (IJIST). Vol.2. No.4. July 2012.
[17] U .Khuranaand, A.Koul. Text Compression and Superfast Searching. Thapar Institute of Engineering and Technology. Patiala. Punjab. India-147004.
[18] V.K.Govindan and B.S. Shajeemohan. IDBE - An Intelligent Dictionary Based Encoding Algorithm for Text Data Compression for High Speed Data Transmission Over Internet.
[19] I. Deslauriers and J. Bajcsy. 2003. Serial turbo coding for data compression and the slepian-wolf problem. in IEEE Information Theory Workshop. August 2003. pp. 296–299.
[20] A. D. Liveris, Z. Xiong, and C. N. Georghiades. 2002. Compression of binary source with side information at the decoder using LDPC codes. IEEE Communications Letters. vol. 6, no. 10. pp. 440–442. October 2002.

AUTHOR’S PROFILE

Debasis Das

Mr. Debasis D. Das completed M.Sc. (Computer Science) from R.T.M. Nagpur University and MCA from Punjab Technical University. He is also a MBA Graduate from Vinayak Mission University. He is pursuing Ph.D. from R.T.M. Nagpur University, Nagpur. He is an Assistant Professor at the Department of Computer Science & Application, VMV Com, JMT Arts & JJP Science College, Nagpur (India). Presently he is working as a research scholar in University Campus, RTM Nagpur University, Nagpur. His research interests include Number System, Cryptography, Data compression and Mobile Technology. He is an associate member of IETE (India).

Dr. U. A. Lanjewar

Dr. U. A. Lanjewar is an Assistant Professor at the Department of Computer Science & Application, VMV Com, JMT Arts & JJP Science College, Nagpur (India). He has around 16 years of teaching and research experience to graduate, post-graduate and doctoral degree students. He has submitted his Post Doctoral Research Work in RTM Nagpur University for Doctor of Science. He has been working as a Research Guide for five universities in the research area of Computer science and Technology, Business Management and Applications and Statistics and six of his students have already been awarded with Doctoral Degree. He has around 50 research papers published in International Peer Reviewed Journals. He has also presented more than twenty research articles in national and international conferences. He has written few books related to his research work and has also worked on various advisory committees of National and International Conferences.