IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-6, Issue-7, July 2018 E-ISSN: 2347-2693

Advanced Lossless Text Compression System based on Dynamic Nibble Reduction Algorithm

Sukhbir Kaur[1*], Paramjeet Singh[2], Shaveta Rani [3]

[1]CSE Department, MRSPTU, Bathinda, India
[2] CSE Department, MRSPTU, Bathinda, India
[3] CSE Department, MRSPTU, Bathinda, India

[*]*Corresponding Author: sukhbirsidhu231993@gmail.com*

Available online at: www.ijcseonline.org

Accepted: 14/Jul/2018, Published: 31/July/2018

Abstract— Data compression is a procedure of shortening the data by following some encoding rules to store or transmit the particular file from one location to another. Data compression is an important task as it cost more to store a large amount of data and to transfer the large amount of data over a network. So we can say that the data compression is a technique by which reduction of data is performed from its original representation so that it can be stored and transferred over the network easily and with lesser cost. In the proposed system, a dynamic nibble based data compression approach is used to compress the data. In this approach data is compressed in the combination of four bits. Performance of the proposed system is evaluated on the various input strings and is compared with the existing systems which is static but reduction algorithm. Performance results shows that the propose system is better than that of the existing system in terms of output bits, saving bits and compression ratio.
.

Keywords—Data compression, Lossless data compression, Nibble based data compression, Static Bit Reduction

I. Introduction

Data compression is a method by that a file (Text, Audio, Video); could also be reworked to a different (compressed) file, specified the first file could also be absolutely recovered from the first file with none loss of actual data. This method could also be helpful if one needs to avoid wasting the space for storing. For example if one needs to store a 4MB file, it should be Preferred to initial compress it to a smaller size to avoid wasting the space for storing.

Also compressed files are way more simply changed over the net since they transfer and transfer a lot of quicker. we tend to need the flexibility to restructure the first file from the compressed version at any time. Information compression may be a methodology of encryption rules that permits substantial reduction within the total variety of bits to store or transmit a file. There a lot of data being restrained, a lot of it prices in terms of storage and transmission prices. In short, information Compression is that the method of encryption information to fewer bits than the first illustration so it takes less space for storing and fewer TRM whereas human activity over a network.

Information Compression is conceivable on the grounds that a large portion of this present reality information is exceptionally repetitive. Information Compression is essentially characterized as a system that decreases the measure of information by applying distinctive techniques that can either be Lossy or Lossless [1].

A. Types of data compression

Currently, two basic categories of information compression are applied in numerous areas. one amongst these is lossy information compression, that is wide accustomed compress image information files for communication or archives functions. The opposite is lossless information compression that is unremarkably accustomed transmit or archive text or binary files needed to stay their data intact at any time.

There are two two categories of data compression:

- Lossy Compression
- Lossless Compression

Lossy data compression

A lossy Data compression strategy is one wherever the data recovers once decompression might not be exactly same because the 1st information, however rather is "sufficiently close" to be valuable for specific reason. once one applies lossy info compression to a message, the message will ne'er be recuperated exactly because it was before it absolutely was compacted. At the purpose once the compacted message is decoded it does not repay the primary message. info has been lost. Since lossy compression cannot be decoded to yield the proper distinctive message, it's something however a good strategy for compression for basic info, parenthetically, literary info. it's most useful for Digitally Sampled Analog knowledge (DSAD). DSAD contains typically of sound, video, illustrations, or image documents. in an exceedingly sound document, as an example, the high and low frequencies, that the human ear cannot hear, may well be truncated from the record.

Lossless Data compression

Lossless info compression could be a procedure that allows the use of data compression calculations to pack the content info and what is more permits the proper distinctive info to be remade from the compacted information. this can be con to the lossy info compression within which the proper distinctive info cannot be recreated from the compacted info. The distinguished nothing record organize that's being used for the compression of data documents is likewise an utilization of lossless information compression approach. Lossless compression is employed once it is very important that the primary info and therefore the decompressed info be indistinguishable. lossless content info compression calculations as a rule misuse factual excess in such a path so as to talk to the sender's info all the additional in brief with no blunder or any quite loss of imperative knowledge contained within the content data info. Since the overwhelming majority of this gift reality info has measurable repetition, on these lines lossless info compression is conceivable. parenthetically, In English content, the letter 'an' is considerably additional typical than the letter 'z', and therefore the probability that the letter 't' are trailed by the letter 'z' is no. thus this kind of excess may be expelled utilizing lossless compression. lossless compression techniques may well be organized by the type of data they're supposed to pack. Compression calculations square measure basically used for the compression of content, footage and sound. Most lossless compression programs utilize 2 varied styles of calculations: one that creates a factual model for the data and another that maps the data to bit strings utilizing this model specified frequently toughened information can deliver shorter yield than improbable(less successive) information.

II. RELATED WORK

In this part various data compression techniques proposed by various authors is presented as below:

U. N. Katugampola,” A New Technique for Text Data Compression” This paper exhibits how ternary portrayal of numbers can be used to pack content information with settled image length coding systems. We utilize a double guide for ternary digits and acquaint a strategy with utilize the parallel 11-sets, which has never been use for coding information, and we additionally utilize 4-Digits ternary portrayal of letter set with lowercase and capitalized letters. We figure out how to limit the length of the bits string, which is just conceivable in ternary portrayal hence productively diminishing the length of the code. We likewise discover some association between this system of coding information and Fibonacci numbers.[1]

A.S. Sidhu,” Text Data Compression Algorithm using Hybrid Approach” , Data Compression is a section that must be nearly attention is text quality assessment. Totally different methodologies are outlined for this purpose. thus selecting the most effective machine learning formula is basically necessary. Additionally to totally different compression technologies and methodologies, choice of a decent information compression tool is most significant. There's an entire vary of various information compression techniques accessible each on-line and offline operating specified it becomes very troublesome to settle on that technique serves the most effective. Here comes the need of selecting the correct methodology for text compression functions and thus an formula which will reveal the most effective tool among the given ones. a knowledge compression formula is to be developed that consumes less time whereas provides a lot of compression quantitative relation as compared to existing techniques. during this paper we have a tendency to represent a hybrid approach to compress the text information. This hybrid approach is combination of Dynamic Bit reduction methodology and Huffman writing. [2].

S. Shanmugasundaram and R. Lourdusamy, “A Comparative Study of Text Compression Algorithms”: There are part of information compression algorithms that are out there to compress files of various formats. This paper provides a survey of various basic lossless information compression algorithms. Experimental results and comparisons of the lossless compression algorithms mistreatment applied mathematics compression techniques and wordbook primarily based compression techniques were performed on text information. Among the applied mathematics writing techniques the algorithms cherish Shannon-Fano writing, Huffman writing, accommodative Huffman writing, Run Length coding and Arithmetic writing are thought of. a collection of attention-grabbing conclusions are derived on their basis. lossy algorithms bring home the bacon higher compression effectiveness than lossless algorithms, however lossy compression is restricted to audio, images, and video, wherever some loss is suitable. The question of the higher technique of the 2, “lossless” or “lossy” is pointless as every has its own uses with lossless techniques higher in some cases and lossy technique higher in others [3].

III. METHODOLOGY

To improve the data compression bits and bits saving percentage for text data, a lossless compression algorithm named as "Dynamic Nibble Reduction Algorithm" is designed. The main idea behind this compression algorithm is to reduce the size of input text by Nibble based approach then further recompress using Huffman coding algorithm

It takes two steps to perform the task of data compression by Dynamic Nibble Reduction algorithm. In the very first step, Dynamic Nibble reduction technique is used to compress the data and in second and final step Huffman algorithm further compress the data obtained in the first step.

In the First step, unique symbols from the input text string are evaluated and then corresponding numeric code are given to these unique symbols. Foe every numeric obtained for these unique symbols a nibble code is generated dynamically to obtain the (compressed) binary output. For every five Nibbles, we generate the equivalent decimal code. This binary output will be than, act as input by converting it into its equivalent ASCII code which is given to the second step as an input.

In the second phase Huffman Coding are applied to the output of initial part to more compress the information and improve the performance of Dynamic Nibble reduction algorithmic program. Huffman committal to writing follows high down approach suggests that the binary tree is made from the highest to all the way down to generate AN best result. In Huffman committal to writing the characters during a file area unit reborn to computer code and therefore the most typical characters within the file have the shortest binary codes, and therefore the characters that area unit least common can have the longest computer code. within the similar manner technique of decompression works in reverse order. Compressed information is initial decompressed by Huffman Decoder and so by Dynamic Nibble reduction decoder to urge back the initial information.

A. Compression Algorithm

Step I: Enter the data required to be compressed.
Step II: Find the unique characters from the given input.
Step III: For every unique symbol find the corresponding numeric code starting from zero.
Step IV: Find the nibble required to represent the character.
Step V: Starting from first symbol in the data notice the Nibble code corresponding to that symbols from assigned numerical codes and concatenate them to get the nibble output.
Step VI: Normalize the nibble output obtained from previous step by adding the required number of zero's to the most significant bit of the output.
Step VII: Get the decimal code for every 5 Nibbles of the output generated by the previous step.
Step VIII: Generate the binary code corresponding to the each decimal value.
Step IX: Provide the generated output from the previous step to adaptive Huffman algorithm to get the final compressed result.
Step X: Display the final result obtained in step VII.
[Step VIII provides the final compressed output]
Step XI: End

B. Decompression Algorithm

Step I: Enter the Final output obtained from compressed phase as an input to the decompression phase.
Step II: Perform the operation of inverse adaptive Huffman coding to generate the numeral codes from the input data.
Step III: Generate the decimal code from the binary code obtained in step II.
Step IV: Generate the nibble code of each decimal obtained from the step III.
Step V: De-Normalize the output obtained from the step IV.
Step VI: Generate the numeral code from the nibble code obtained in step V.
Step VII: Extract the unique symbol for every numeral value from the symbol table and replace it the corresponding numeral value.
Step VIII: Combine results obtained in the previous step for every numeral values and obtain the final decompressed output.
Step IX: Show the final output to the user.
Step X: End.

IV. RESULTS AND DISCUSSION

In this part, results generated by the proposed system are discussed. System takes various different inputs to test the performance of the system.

Performance Parameters: Two parameters are used to evaluate the performance of the proposed system which are Compression Ratio(CR) and Bits Saved.

1. **Compression ratio:** Compression ratio can be defined as the ratio of size of the uncompressed file to the size of the compressed file.

$$\text{Compression ratio} = \frac{\text{INPUT BITS}}{\text{OUTPUT BITS}}$$

2. **Bits Saved:** It is the difference between the uncompressed bits and compressed bits.

Bits Saved = Uncompressed bits -Compressed bits

Table 5.1: Result comparison of Static bit and Dynamic Nibble Compression algorithm on the basis of output bits

Sr. No.	No Of String	Input Size	Existing Output (Bit)	Proposed output (Bits)	Difference in existing & proposed system (Bits)
1	String 1	696	208	120	88
2	String 2	936	288	200	88
3	String 3	1408	696	220	476
4	String 4	2816	1392	450	942
5	String 5	4288	2288	832	1456

As it can be seen from the above table the proposed system compress more as compared to the existing system. It can also be seen that as number of input bits increases the difference in the existing and proposed output bits also increases.

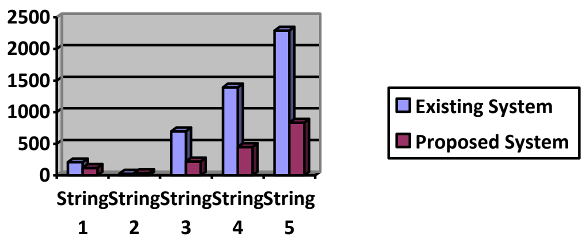

Fig 5.1 Comparison graph of the existing and proposed system on the basis of output bits .

Table 5.2: Result comparison of Static bit and Dynamic Nibble Compression algorithm son the basis of bit saved

Sr. No.	No Of String	Input Size	Existing Bits Saved	proposed Bits Saved
1	String 1	696	488	576
2	String 2	936	648	736
3	String 3	1408	712	1188
4	String 4	2816	1424	2366
5	String 5	4288	2000	3456

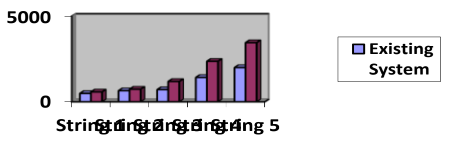

Fig. 5.2 Comparison graph of the existing and proposed system on the basis of Bits saved .

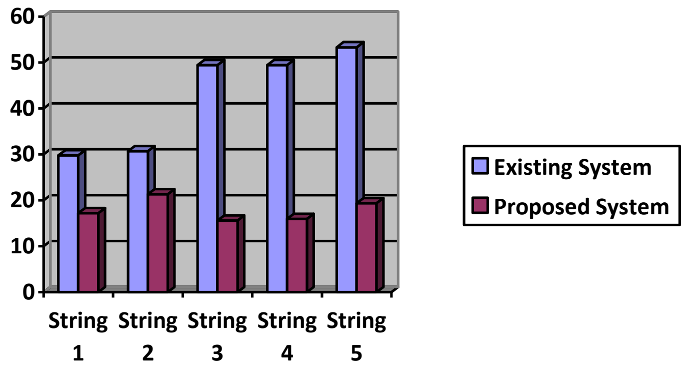

Fig. 5.3 Comparison graph of the existing and proposed system on the basis of compression ratio.

V. Conclusion and Future Scope

In this proposed work, a Dynamic Nibble reduction algorithm is developed to compress and decompress the text data based on lossless data compression approach. Various experiments are conducted on totally different datasets appreciate Random, alphabetical, Numeral and Special Characters dataset. The results obtained by the planned system square measure compared with the present information compression techniques- Static Bit Reduction and Huffman cryptography victimization parameters- Compression quantitative relation and Saving Bits and output bits.

From the results analysis, it's complete that the planned system shows excellent compression ends up in terms of Compression quantitative relation, Output bits and Saving share as compared to the present techniques for all the datasets that are thought-about.

The existing Static Bit reduction system provides poor compression results. it's supported mounted bit cryptography theme and provides lossy output if special characters square measure gift within the computer file. These limitations of the present Static Bit reduction system are overcome by the planned system because it is victimization variable Bit cryptography theme. The compression results shown by the planned system square measure higher than the present systems (Static Bit reduction and Huffman cryptography) because it is victimization dynamic Nibble reduction technique within the 1st section and Huffman coding is applied within the second section to additional improve the performance of the planned system and to attain higher compression results

	No Of String	Input Size	Existing CR	Proposed CR
1	String 1	696	29.8	17.24
2	String 2	936	30.7	21.36
3	String 3	1408	49.4	15.62
4	String 4	2816	49.4	15.98
5	String 5	4288	53.3	19.40

Table 5.3: Result comparison of Static bit and Dynamic Nibble Compression algorithm son the basis of compression ratio

FUTURE WORK

Dynamic Nibble Reduction algorithmic rule works solely with text information written in single language which might even be tested to compress the multi lingual information i.e. text information written in multiple languages during a Indian file. In alternative words, the system works solely on American Standard Code for Information Interchange information set which might be extended to figure with Unicode data for future work.

REFERENCES

[1] U. N. Katugampola, "*A New Technique for Text Data Compression*",International Symposium on Computer, Consumer and Control , 978-0-7695-4655-1 pp. 405-409, 2012.

[2] A.S. Sidhu, M. Garg, "*Text Data Compression Algorithm using Hybrid Approach*", International Journal of Computer Science and Mobile Computing, Vol.3 Issue.12, pp. 01-10, 2014.

[3] S. Shanmugasundaram and R. Lourdusamy, "*A Comparative Study of Text Compression Algorithms*",International Journal of Wisdom Based Computing, Vol. 1 (3), pp 68-76 , 2011.

[4] S. Kapoor and A. Chopra, "*A Review of Lempel Ziv Compression Techniques*" IJCST, Vol. 4, Issue 2, 2013.

[5] M.A.D. Suarjaya, "*A New Algorithm for Data Compression Optimization*", International Journal of Advanced Computer Science and Applications, Vol. 3, No. 8, pp.14-17,2012.

[6] S.R. Kodituwakku and U. S. Amarasinghe , "*Comparison Of Lossless Data Compression Algorithms For Text Data*" Indian Journal of Computer Science and Engineering, Vol. 1, No. 4, pp. 416-425 , 2010.

[7] R. Kaur and M. Goyal, "*An Algorithm for Lossless Text Data Compression*" International Journal of Engineering Research & Technology , Vol. 2 Issue 7, 2013

[8] H. Altarawneh and M. Altarawneh, "*Data Compression Techniques on Text Files: A Comparison Study* ", International Journal of Computer Applications, Vol. 26 No.5, 2011.

[9] U. Khurana and A. Koul, "*Text Compression And Superfast Searching*" Thapar Institute Of Engineering and Technology, Patiala, Punjab, India-147004

[10] A. Singh and Y. Bhatnagar, *"Enhancement of data compression using Incremental Encoding*" International Journal of Scientific & Engineering Research, Vol. 3, Issue 5,2012.

[11] A.J Mann, *"Analysis and Comparison of Algorithms for Lossless Data Compression*", International Journal of Information and Computation Technology, Vol. 3, No.3, pp. 139-146, 2013.

[12] K. Rastogi, K. Sengar, "*Analysis and Performance Comparison of Lossless Compression Techniques for Text Data*" International Journal of Engineering Technology and Computer Research, Vol. 2 (1), pp. 16-19, 2014.

[13] M. Sharma, "*Compression using Huffman Coding*", International Journal of Computer Science and Network Security, Vol.10 No.5, 2010.

[14] S. Shanmugasundaram and R. Lourdusamy, "*IIDBE: A Lossless Text Transform for Better Compression* " International Journal of Wisdom Based Computing, Vol. 1 (2), 2011

[15] P. Kumar and A.K Varshney, " *Double Huffman Coding* " International Journal of Advanced Research in Computer Science and Software Engineering , Vol. 2, Issue 8, pp. 517-520, 2012.