International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435 (Online) Volume 15, Issue 07, July 2026)

Dictionary Based Data Compression and Pattern Searching in the Compressed Data

Sukhvinder Singh Bamber
University Institute of Engineering & Technology, Panjab University SSG Regional Centre, Hoshiarpur, Punjab, India

***Abstract* — Effective data compression is the primary issue with data management techniques. One of the most well-liked algorithms between several compression methods is the LZW compression method. Compressed pattern matching raises additional issue. The difficulty of finding patterns in compressed data and reporting all instances of the patterns without decompressing the compressed data is tackled by the major study field of compressed pattern matching. A new data compression approach called "Compression data Based on Dictionary" & "Searching Algorithm in Compressed Data" have indeed been designed, analysed, and tested in this thesis. This algorithm generates the word compression codes and keeps a dictionary that may be used for both compression and decompression. For discovering the necessary results, the search algorithm makes use of a quick search word matching method.**

***Keywords* — LZW compression, compression, Compressed Pattern matching, Dictionary, Quick Search.**

I. Introduction

Now the question of the compression is now raised. Compression is the process of shrinking data to reduce its size and speed up transmission. Depending on a number of variables, compression can be applied to either the data content alone or to the full transmission unit (including header data) [1]. Two broad categories of compression exist:

A. Lossless Compression

The same original data can indeed be reconstructed from the compressed data using a category of data compression techniques called lossless data compression [1].

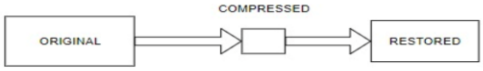

Figure 1: Lossless Data Compression

B. Lossy Compression

In lossy compression some of the information from the original message sequence is lost and the original sequences cannot be regenerated from the compressed sequence [2].

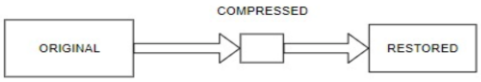

Figure 2: Lossy Data Compression

II. Literature Survey

There are numerous different compression techniques, some of which are listed here based on the types of compression, namely lossless and lossy compression. Dictionary Codes (Zip File Format and LZW Data Compression), Entropy Encoding (Huffman Coding, Shannon Fano Coding, and Arithmetic Encoding), and Run Length Encoding are methods used in lossless compression. Scalar and vector quantization are methods for lossy compression. one another method of lossy compression is transformed coding. This method focuses on the LZW compression technique, a lossless type of compression. [4,5,6,7,8,9,18,11,12,13,14].

A. Lempel and J. Ziv named LZW compression, which Terry A. Welch later modified. Due to its simplicity and adaptability, it is the preferred technique for general-purpose data compression. Typically, LZW will cut text, executable code, and accompanying data files to around half their original size [15].

A substitution or dictionary-based encoding scheme is what LZW is classified as. The algorithm creates a data dictionary, often referred to as a translating table or string table, of the material gathered in an uncompressed stream of data. Frequent patterns (substrings) are discovered inside the stream of data and matched to dictionary entries. If the substring is not already present in the dictionary, a code expression based on the substring's data content is generated and added to the dictionary. The compression output stream is then written with both the expression. [16].

To get search something in the compressed data booyer and moore gave an algorithm which is discussed [17] and various new optimised techniques to find the pattern in the compressed string so that all the searching related work can be done easily over limited resources [18,19,20,21].

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435 (Online) Volume 15, Issue 07, July 2026)

We do have the various other algorithms which do not depend upon the mixed complexities which required by the algorithms like the LZW [22] and the dictionary-based data of the new compression and the searching of the data [23] where you don't have to thorough about the technique which so much optimized by the machine learning trained model of the data. [24,25,26].

If we are considering the compression, then the decompression is also equally important as the compression. But in case of the loosely compression decompression is not possible because some part of the data is lost but if you are doing a compression which is lossless you can have the data all decompressed that was present initially [27]. Techniques which were used in the lossless compression were discussed in the details [28,29,30].

In this paper we are doing a thorough comparison between the LZW compression and the Dictionary based data compression algorithm compression is a very old and very widely used data compression technique which is built over the lossless data compression of the data. There are merits of using the LZW compression but there are various de merits of using it.

In this thesis our prime goal is to examine a algorithms which is best in all the scenarios of the data compression and searching. In the beginning we have started analysing various algorithms and techniques to compress data and perform searching over it.

Two very popular algorithms which were used everywhere, and they have very wide usage in the industry which are Huffman coding and the LZW compression. For the smaller files of the data we use the Huffman coding while for the big amount of the data files we used the LZW compression algorithm both of them are the part of the lossless data compression.

The prerequisite for the data compression is only that you have well knowledge of how data can be compressed with the help of the data compression techniques. what are the types of the data compression and how they are used in the world of the fast paced data searching tools of the media sharing world.

Reader needs to know what are the two types of the data compression which were discussed in this research paper and their differences that how lossy is different from the lossless data compression.

Practical uses of the lossless data compression and what is the difference between the LZW and dictionary based compression.

In terms of reducing file sizes, Dictionary Based Compression Algorithm is discovered to be superior to LZW.

While compressing text files of a bigger size, dictionary-based compression performs better; when compressing smaller text files, LZW compression comes to mind. A pattern can be found in a compressed file using a dictionary-based compression algorithm without having to first decompress it. Because the search was done in a compressed file, the search time was reduced. It would then be demonstrated as the best concept for looking for patterns in compressed text files. This compression and searching method are only useful for text-based material. Any sort of data, including image data, is not supported.

These are the things which needs to be taken care of when we are dealing with the data compression and analysis of the data compression algorithms.

III. System Description

The hardware or the system requirement to run both the Lempel Ziv welch and the dictionary based data compression algorithms is not so fancy. The first and the foremost thing to run the algorithms is a Unix based operating system.

In the Unix or Linux based operating system we have various open source tools which makes Our comparative based study more easier for the diagrammatic side we have used various open Source software to create very enhanced and the rich diagrams.

LZW and the dictionary based data compression require 500 MB of free data in the hard disk or any other secondary storage to store the data and minimum 2 gigs of the random access memory to perform this comparison in addition to all of this we require or these particular algorithms require native local environment setup to compress the data because some algorithms performs better in the python language but algorithms like Huffman coding or algorithmic encoding performs better in the c plus plus language so for the languages like c plus plus we require GNU gcc compiler while the languages like python we require python 3 compiler cum interpreter.

In this paper we are comparing text only, but you can compress and search images as well as the videos of the different format in the like 3gp, mp4 etc. for this you require one additional thing in your computer which is graphic card.

IV. Results And Discussions

The LZW Compression Algorithm provides a way to reduce larger file sizes, and it has been shown in practise that its performance shouldn't be underestimated.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435 (Online) Volume 15, Issue 07, July 2026)

Dictionary Based Compression Algorithm, the proposed algorithm, is found to be superior to LZW in terms of file size reduction in the practical analysis. Here, performance analysis of both algorithms is explained using comparative data, and the evaluation is based on file sizes:

A. Compression Result Analysis between LZW Compression and Dictionary Based Compression

Smaller File Size:

S. No.	Text File Size	LZW Compressed File	Dictionary Based Compressed File
1	50	60	70
2	100	85	75
3	200	100	77
4	300	110	105
5	400	120	115

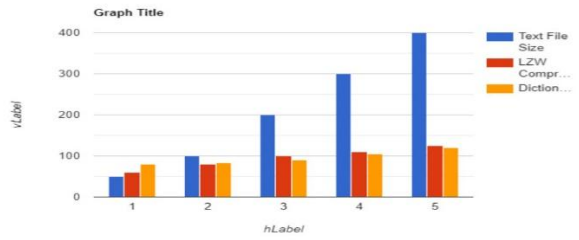

Figure 1

In the "Comparative Table" of figure 1, different file sizes are taken in (KB's) having different data sets, to be compressed i.e., relatively small size file as it will be compared with the larger file sizes. And the same file is compressed with both algorithms which is already being developed and tested. Clearly it can be analyzed with the help of bar graph i.e., "Comparative Figure 1" that the text.

Both methods resulted in smaller file sizes, but it has been noticed that they occasionally function equally well and that dictionary-based compression results in smaller files. It's possible that establishing a dictionary will increase the compressed file size if the original file size is quite small. The likelihood of receiving the same compressed file size may also exist if the file size is discovered to be extremely small, which would reduce the effectiveness of shrinking file sizes. But as the file size grows, the dictionary-based technique's performance at reducing file size improves as you start comparing it to the LZW compression algorithm. As a result, it might not be better for smaller file size.

Larger File Size:

S. No.	Text File Size	LZW Compressed File(in MB's)	Dictionary Based Compressed File(in MB's)
1	25	9	6
2	45	15	12
3	65	20	17
4	85	35	30
5	100	40	37

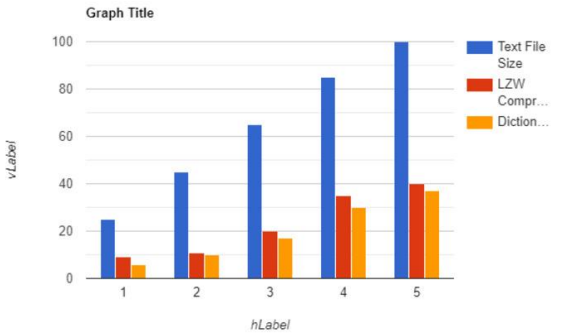

Figure 2

Although the heterogeneous datasets in "Comparative Table" of figure 2 have various file sizes again, the file sizes are still considerably bigger than those in the prior discussion. This algorithm was created specifically to compress large files, and it can be understood by referring to the bar graph in "Comparative Figure 2," which shows that the dictionary-based algorithm produces significantly smaller files than the LZW compression algorithm. The performance of creating an algorithm gradually improves with greater file sizes. It is discovered to be more efficient, therefore it is obvious that even if the file is too big to be compressed, the developed technique would always prove to be better in terms of compression and compressed file size.

B. Searching Analysis in Compressed File using Quick Search String Matching Algorithm

The ability to look for a pattern in a compressed file without having to decompress it is the strength of the dictionary-based compression algorithm. Because the search was done in a decompressed file, the search time was reduced. It would then be shown as the best concept for looking for patterns in compressed text files.

Because the code of the pattern to be searched is not found in the dictionary, there is no need to search it in the compressed text file, this designed technique would function amazingly well when the pattern is not discovered in the text file. With the exception of all other searching approaches, this improves the performance of pattern searching overall.

In general, if a pattern is to be searched in a compressed file, the pattern's code is first checked against a dictionary; if a match is found, the search will only begin in the compressed file after it has been loaded using a quick search algorithm, which can locate all patterns in the compressed text file. This is how pattern searching is optimized using developed algorithms.

V. Conclusion

In terms of reducing file sizes, Dictionary Based Compression Algorithm is discovered to be superior to LZW. While compressing text files of a bigger size, dictionary-based compression performs better; when compressing smaller text files, LZW compression comes to mind. A pattern can be found in a compressed file using a dictionary-based compression algorithm without having to first decompress it. Because the search was done in a compressed file, the search time was reduced. It would then be demonstrated as the best concept for looking for patterns in compressed text files. This compression and searching method are only useful for text-based material. Any sort of data, including image data, is not supported.

REFERENCES

[1] N. Okamoto, S. Kimura, Y. Ebihara, "An introduction of compression algorithms into SSL/TLS and proposal of compression algorithms specialized for application", In proc. IEEE Conference, March 2009, pp. 817-820.

[2] Patauner, A. Marchioro, S. Bonacini, A. Ur Rehman, W. Pribyl, "A lossless data compression system for a real-time application in HEP data acquisition", In proc. Of 17th IEEE-NPSS Conference , May 2010, pp. 1- 6.

[3] S. Jalali, T. Weissman, "Denoising via MCMC-Based Lossy Compression", IEEE Transactions on Signal Processing, June 2012, vol. 60, pp. 3092-3100.

[4] T. Valencia, L.O. Cerdeira, E.L. Iglesias, F.J. Rodríguez, "Translation table compression under EndTagged Dense Code", In proc. 4th International Conference on Universal Communication Symposium (IUCS), Oct. 2010, pp. 306-311.

[5] A.W. McMorran, A.N. DeVos, J.P. Britton, G.W. Ault, "ZCIM: A compressed, modular CIM data exchange format", July 2010, pp. 1-5.

[6] D.J. Jackson, S.J. Hannah, "Comparative analysis of image compression techniques", In proc. Of Twenty-Fifth Southeastern Symposium on System Theory, Mar 1993, pp. 513-517.

[7] M. Ezhilarasan, P. Thambidurai, K. Praveena, S. Srinivasan, N. Sumathi, " A New Entropy Encoding Technique for Multimedia Data Compression", In proc. Of International Conference on Computational Intelligence and Multimedia Applications, Dec. 2007, pp. 157-161.

[8] Lee Taeyeon, Park Jaehong, "Design and implementation for static Huffman encoding hardware with parallel shifting algorithm", IEEE Nuclear Science Symposium Conference Record, Oct. 2003, Vol. 2, pp. 1314-1318.

[9] T. Tjalkens, "Implementation cost of the HuffmanShannon-Fano code", In proc. Of Data Compression Conference, March 2005, pp. 123-132.

[10] Yuh - Ming Huang, Yin-Chen Liang, "A secure arithmetic coding algorithm based on integer implementation",", In proc. 11th International Conference on Communications and Information Technologies (ISCIT), Oct. 2011, pp. 518-521.

[11] S.D. Bradley, "Optimizing a scheme for run length encoding", Proceedings of the IEEE, Vol. 57, Issue 1, Jan. 1969, pp. 108-109.

[12] Barnlund, D. (2008). A transactional model of communication. Communication Theory. 47- 57.

[13] Jun Fang, Hongbin Li, "A study of hyperplane-based vector quantization for distributed estimation", In Proc. IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP), March 2010, pp. 2898-2901.

[14] Guy E. Blelloch" Introduction to Data Compression" "http://www.cs.cmu.edu/~guyb/realworld/compression.pdf

[15] "The Scientist and Engineer's Guide to Digital Signal Processing", by By Steven W. Smithhttp://www.dspguide.com/ch27/5.htm

[16] Tao Tao, Student Member, IEEE, and Amar Mukherjee, Fellow, IEEE, Pattern Matching in LZW Compressed Files IEEE TRANSACTIONS ON COMPUTERS, VOL. 54, NO. 8, AUGUST 2005.

[17] Boyer, R. S.; Moore, J. S. A fast string searching algorithm. Communications. ACM 1977, 20, 762-772.

[18] Xiangling Kuang, Guangqiu Huang, "An optimization algorithm based on ant colony algorithm", In the proc International Conference on Automatic Control. And Artificial Intelligence (ACAI 2012), March 2012, pp. 469-472.

[19] Sunday, D. M. A very fast substring search algorithm. Commune. ACM 1990.

[20] Fast pattern matching in strings, D. E. Knuth, J. H. Morris, Jr and V. B. Pratt, SIAM J. Computing, 6, 1977, pp. 323–350

[21] M. Charikar, E. Lehman, Ding Liu, R. Panigrahy, M. Prabhakaran, A. Sahai, and A. Shelat, "The smallest grammar problem," IEEE Transactions on Information Theory, 2005.

[22] J. Ziv and A. Lempel, "Compression of individual sequences via variable-rate coding," IEEE Transactions on Information Theory, 1978.

[23] T. A. Welch, "A technique for high-performance data compression," Computer, 1984.

[24] N. J. Larsson and A. Moffat, "Off-line dictionary-based compression," DCC, 1999.

[25] G. Nevill-Manning and I. H. Witten, "Compression and explanation using hierarchical grammars," The Computer Journal, 1997.

[26] Hucke, M. Lohrey, and C. Philipp Reh, "The smallest grammar problem revisited," in SPIRE, 2016.

[27] J. Esparza, P. Rossmanith, and S. Schwoon, "A uniform framework for problems on context-free grammars," Bulletin of the EATCS, 2000.

[28] K. Thompson, "Programming techniques: Regular expression search algorithm," Communications of the ACM, 1968.

[29] Navarro, "Regular expression searching on compressed text," Journal of Discrete Algorithms, 2003.