Indonesian Journal of Electrical Engineering and Computer Science
Vol. 16, No. 2, November 2019, pp. 890~896
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v16.i2.pp890-896

A new compression technique in MANET: compressed-LZW algorithm

Hadeel Noori Saad[1], Fairouz mushtaq Jafar[2], Hanan Abbas Salman[3]
[1,2]Department of Computer, Faculty of Education for Women, University of Kufa, Iraq
[3]Najaf Technical Institute, Furat Al-Awsat Technical University, Iraq

Article Info

Article history:

Received Jan 29, 2019
Revised Apr 30, 2019
Accepted May 21, 2019

Keywords:

Coding
Consumption
Energy
Energy aware
LZW compression
MANET

ABSTRACT

Now a days mobile ad hoc networks MANETs play a vital role in tremendous communication applications. This networks require infrastructure less module in non- centralized application's area. Low battery power is of great concern to sustain for a long time, thus researchers try to assist in this critical maintenance point. This work proposed an enhanced version of well-known compression technique LZW to reduce data communication and maintain battery life. Compressed-LZW is a technique to code data stream into further compact form by merging successive redundant compressed code, and achieved by padding the code stream with the displacement of the next similar code. Compressed-LZW outperform its successor in saving consumed energy as expressed in the implementation and results by generating less output codes.

Corresponding Author:

Hadeel Noori Saad,
Department of Computer, Faculty of Education for Women,
University of Kufa, Iraq.
Email: Hadeel.jrew@uokufa.edu.iq

1. INTRODUCTION

The wide spread application area of communication inspired the needs to light, self-organized, and mobile networks [1]. This gives MANET its reputation and researcher's interest. Many researches investigate its abilities and methods to enhance its limitations. MANET inadequacies mitigations lie on finding strong cryptosystems, enlarging transmission range and controlling battery drain, the former covered by optimize impenetrable crypto-algorithms, topology engineering and routing protocol optimization [2-3]. On the other hand link failure reduction and route reconfiguration, communication minimization, data compression techniques, and power control mechanisms, are the main fields of battery life enhancement. Such optimization techniques should take processing time, and resources availability into consideration too.

MANET routing protocols maintain optimized routing with minimum cost and controlling overheads. AODV, DSR, DSDV and other ad hock routing protocol's route maintenance mechanisms focus on finding and remember alternative routes for further computation elimination and efficient link failure recovery [4-7]. MAC layer local information participate in power control management through sleep-wake up modes [8-10] which prolong the battery life. Idle ineffective nodes can go to sleep mode, where the wakeup can be made through local information, beacon control messages, and higher layers on demand information.

This work focus on reduce communications via data compression upon new LZW compression technique. The proposed technique work as two level compression, where the successive redundant codes resulted from LZW algorithm are further merged and flagged. The following subsections will exploit the literatures related to compression techniques, LZW compression, the proposed technique in details. An excessive implementation and simulation results will be discussed. Finally conclusion section will give the main conclusion points derived from the results.

2. LITERATURE REVIEW

Data compression can reduce the number of transmitted bits and utilize battery power. Lossy and lossless compression are the two broad categories under which a wide variety techniques had been proposed. Sound and images can benefit from Lossy techniques in spite of the data degradation where best effort are still enough. Whereas, text files use lossless technique to obtain an identical to original data. Header, payload, and bulk compression is another compression classification schemes accordingly variety of compression standards had been developed. In this section we will cover header and payload schemes, while bulk compression will be discussed at the next one.

The compressed transport protocol (CTCP) header compression was developed by V. Jacobson [11]. It describes a method for compressing packet headers of IPv4/TCP. The compression depend on building flow context ID on both sides, it gain 10% compression which improve the performance on low speed serial links. Losing single packet can result on successive packet deletion result from asynchronous context id. IEFT working group proposed ROHC [12] header compression scheme, it is particularly suitable for wireless communication. ROHC scheme yield bandwidth savings up to 60%. Its advantage over VJ compression, that it work effectively on communication with high error rates. This scheme use context id and represent the compression process as evolution across three states, so it classified under stateful schemes. Any feedback error can reverse the flow to lower. Successive versions of ROHC supports many protocols,eg. TCP, ESP, RTP, UDP, UDPLite, and IP protocols [13-16].

Jeannot et al.,[17] proposed an adaptive online compression algorithm AdOC suited for online application data transfer. It automatically adapts the level of compression to the speed of the network. It divide the compression process to Multithreading, compression and communication. Compression thread push data to FIFO data buffer, while communication thread pop it from the buffer. AdOC algorithm compresses data into smaller chunks. It works well with long run network changes.

Payload compression is another mode of compression schemes which perform the compression on payload fields of packets. On-the-fly compression proposed by Krintz and Sucu in [18] is an adaptive Compression Environment work upon forecasting network's resources performance which adapt to. On-the-fly selects the a compression scheme among several depend on forecasting result, it proven to improve transfer performance by 8-93 percent. Despite it may be profitable on underlying resource performance, its forecasting process will add more overhead computation time.

6LoWPAN (IPv6 over low power wireless area networks) [19] compression uses known or assumed headers in the compression process and thus it differs from ROHC, where there is no need to create and synchronize decompression tables at the receiver side and work in stateless manner. This has an advantage in wireless lossy communication. 6LoWPAN can achieve IP6 48 and 60 byte header to 7 and 7-31 byte compression ratio respectively.

R. Maulunida and A. Solichin [20] Introduce variable-length-coding to form an optimized LZW dictionary, the proposed algorithm found successive matches and combine them with longer patterns, this give 38.35% compression ratio.

2.1. LZW Compression Technique

One of the widely used lossless data compression is LZW, developed by A. Lempel and J. Ziv, and Terry A. Welch [21] it is a dictionary-based algorithm. LZW algorithm works best for highly repetitive data. LZW compression is fast comparing to other compression algorithms. This algorithm is an updated version of LZ78 algorithm published in 1978 (LZ78) [22]. Zive was published its origins in 1977 and it is named as LZ77 [23]. It is known as LZ2 and LZ1, respectively. The LZW algorithm uses index and dictionary for encoding- decoding operation. It creates a dictionary and data will be examined against the dictionary, if a match is found, then corresponding string is replaced with the index. In [24] proposed an optimized LZW using binary search which fasten the dictionary-matching process. The experiment in [25] shows improvement on compression 94.41% and 93.34% on decompression. Amit Setia and Priyanka Ahlawat proposed an enhanced LZW (ELZW) algorithm. Because most of data files we work with contains frequent spaces, ELZW algorithm eliminates those spaces from the data files. The algorithm expose an efficient behavior, results in high compression ratio. Figure 1 shows LZW compression technique flowchart.

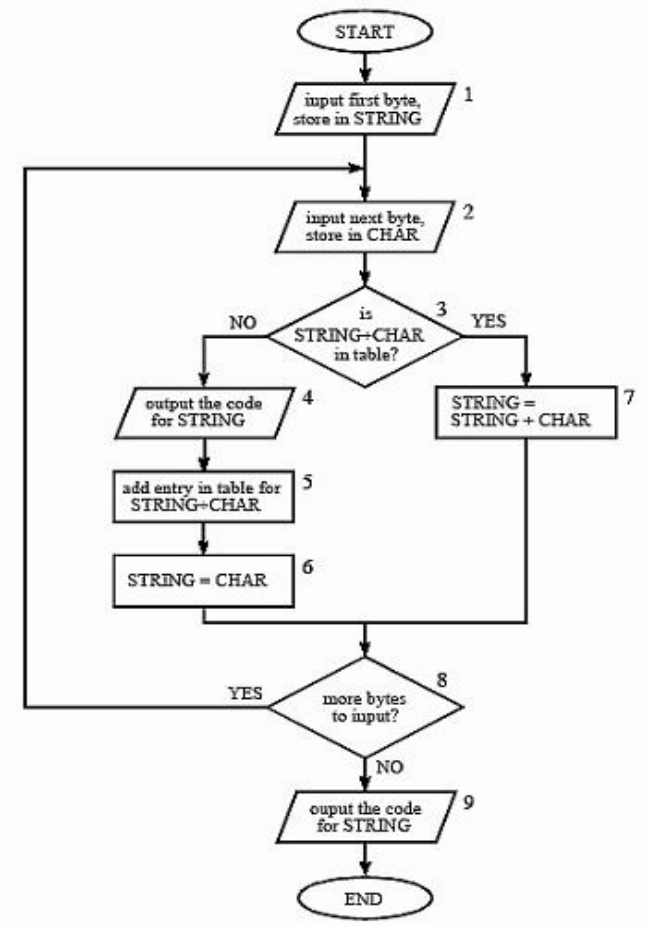


Figure 1. LZW compression technique

3. RESEARCH METHOD

In this section the implementation and simulation methods are discussed. The new algorithm provide two coding scheme, 9-bit and 12-bit scheme for 8-bit character coding (or 17-bit and 20-bit for 16-bit character coding), where each LZW-coded data will tagged with one-bit flag to point to one of the two code schemes. Using such coding scheme will be able to compress each two successive redundant 16 bit (two 8-bit characters) into 12-bit code and each 32 bit (two 16-bit character) into 20-bit code. It records the occurrence of the next existing character and compute its displacement from the current position (index). The 3-bit displacement then will be concatenated with the current character code, the algorithm provide data reduction for redundant data within 8 byte forward displacement. Table 1 express the proposed compressed-LZW algorithm. Proposed coding scheme as shown in Figure 2.

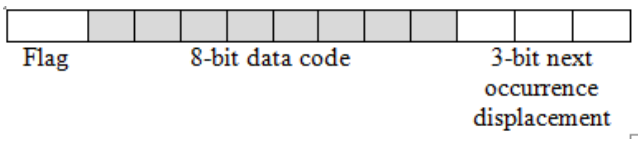


Figure 2. Proposed Coding Scheme

In Algorithm 1, steps 9 and 10 are the core optimization of the new algorithm, it depend on the flag of the current entry to compute the displacement of the next similar code occurrence, and pad the output with. This will merge successive couple redundant data into single output character of reduced bit length as mentioned previously.

Algorithm 1. Compressed-LZW

```
Proposed LZW PSEUDOCODE
 1 Initialize table with single character strings
 2 Initialize coding list to empty
 3  P = first input character
 4  WHILE not end of input stream
 5    C = next input character
 6    IF P + C is in the string table
 7     P = P + C
 8     If current entry's    flag==false
 9      get previous-index of the occurrence of P+C to the coding list
 10       Set current previous-index to  the index of current input
 11      current string table entry's  flag=true
       end if
 10    ELSE
 11     output the code for P
 12     add P + C to the string table
 13     P = C
     End if
 14   END WHILE
 15   output code for P
```

3.1. Implementation

This Section will explore the implementation and comparison of Compressed-LZW and the standard LZW algorithm, with different data patterns. Using different data patterns will reveal weak and strength points of the new algorithm. Table 1 exhibits the compression scheme after executing the proposed algorithm. The implemented character string in Table 1 is the same as the one in [25] for comparison purpose.

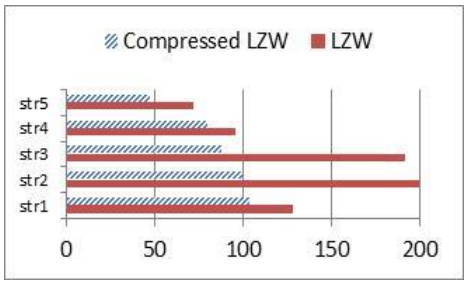
Table 1. Implementation and Comparison of Compressed LZW Algorithm

S no.	Char	String+ Char	In Table	LZW Output	Compressed-LZW Algorithm	Add to Table	New String
1	A	A					A
2	B	AB	NO	A	A	256=AB	B
3	C	BC	NO	B	B	257=BC	C
4	A	CA	NO	C	C	258=CA	A
5	B	AB	YES(256)				AB
6	B	ABB	NO	256	**12 bit , 256 + code**	259=ABB	B
7	A	BA	NO	B	B	260=BA	A
8	B	AB	YES(256)				AB
9	B	ABB	YES(259)				ABB
10	B	ABBB	NO	259	259	261=ABBB	B
11	C	BC	YES(257)				BC
12	A	BCA	NO	257	257	262=BCA	A
13	B	AB	YES(256)				AB
14	A	ABA	NO	256		263=ABA	A
15	B	AB	YES(256)				AB
16	B	ABB	YES(259)				ABB
17	B	ABBB	YES(261)				ABBB
18	D	ABBBD	NO	261	261	264=ABBBD	D
19	EOF	D	NO	D	D		

Through the comparison with the standard LZW as shown in Table 2, the output at row 14 was encoded and merged with previous code occurrence (in line 6) then the compressed-LZW outperform the standard algorithm (using 9 output characters) by 0.1 through single character reduction.

To examine the new algorithm against different data patterns and analyze the effects on its performance, the implementation was included with 5 different patterns as shown in Figure 3(a). Figure 3 gives the data analysis for compression rate for both algorithms.

Str1:
abcdabcdabcdabcdabcdabcdabcdabcdabcd
Str2:
abcdabddabccabcaabbdabbaabdaabcbbbcddbcd
Str3:
abcdabcdabcdabcdabcddcabbdacbddccaaabcacb
Str4:abababababababababababababababababab
Str5:aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa



(a) 3(b)

Figure 3(a) Character streams. (b) Standard and compressed LZW compression ratio

The best behavior of the new algorithm is at str5 as shown in Figure 3(b) where the data is of single redundant alphabet, whereas, str2 and str3 reveal the drawback of LZW compression. These two strings contain redundant data in random manner.

Figure 4 show the implementation of the Compressed-LZW with 8 different file sizes ranging from 12900 to 103200, (multiples of 12900) characters, each file contain random data. As shown, the new algorithm can gain compression rate better than the standard LZW algorithm, where the former can generate 15100 output characters for the 103200 character entry, while the later gained 21000 character.

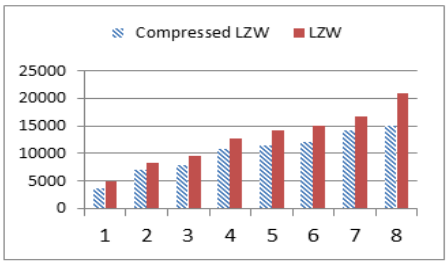


Figure 4(a). input-output character stream

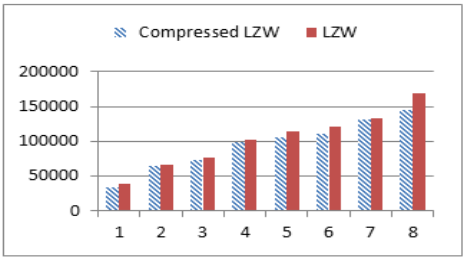


Figure 4(b). 8-bit code

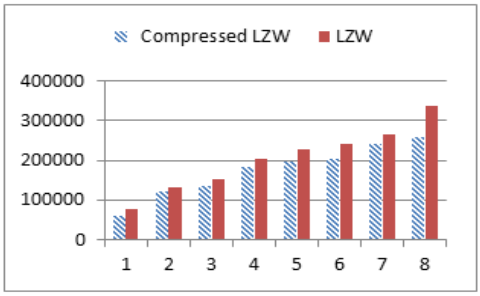


Figure 4(c). 16-bit code

Figure 4(a-c). Implementation of compressed-LZW and standard LZW on data files

3.2. Simulation and Experiment

The compressed-LZW was proposed as an energy aware solution, through communication reduction. In this work the algorithm had been implemented on MANET with 10 nodes on NS 2.3 simulator. Throughout the simulation period node1 (source node) registers the energy consumption for later comparison purpose with LZW compression algorithm. Three simulation scenarios had been designed and executed, raw data (without compression), Compressed-LZW compression, and LZW compression respectively. Table 2 show NS2.3 simulation environment's parameters.

Table 2. Simulation Environment

No. of Nodes	10
Initial energy in joules	1000
Routing Protocol	DSDV
Transmission Range	500
Packet size	512

As shown in Figure 5 which exhibit the residual energy after 1000 sec. simulation period. The Compressed- LZW as shown in Figure 5 would save the battery energy with 10% over LZW algorithm, using 8-bit coding, and 16-bit coding.

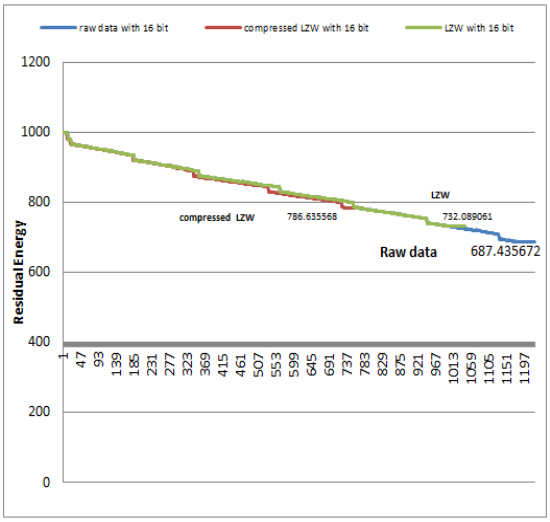


Figure 5(a)

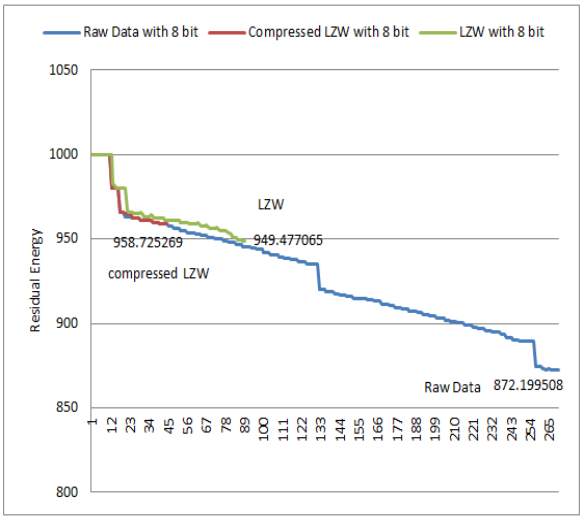


Figure 5(b)

Figure 5. The residual energy after 1000 sec. simulation period, Figure 5(a) compression with 16-bit coding Figure 5(b) compression with 8 bit coding

4. RESULTS AND DISCUSSION

As shown in the previous implementation section, the compressed LZW work very well with a data which is rich of redundant patterns as in Figure3(b) where the performance of this algorithm was obviously revealed. Str5 in in Figure3(a) comprise a single redundant character this give high chance to couple each two successive characters into single output this can increase the compression ratio to more than 50%. the power of the algorithm also increased with long binary coding, where the effect of the padded index tag at the rear 3-bit displacement vanished relatively with long bit stream as shown in in Figure4 (b and c) respectively.

In the simulation section Figure 5 shows the residual energy enhancement by reducing the number of output characters, through the reduction of communications between nodes. The simulation registration showed the residual energy after implementing the two compression algorithms as 949.477065 and 958.7253 joule for 8-bit code and for 16-bit code 732.089061, and 786.2134 for LZW and Compressed-LZW, respectively. These results reflect the strengths of the compressed-LZW in saving the energy and the network life at all.

5. CONCLUSION

Compressed LZW implement further reduction through successive similar codes, it work as layer 2 compression for the standard LZW (layer 1). This algorithm based on tagging each output to classified the output to merged data or not. The first require the decoder to read the rear of the code to know the displacement of the merged code for regeneration purpose.

The implementation show outperformance of compressed algorithm over LZW, it saved 10% of node's energy. Compressed LZW works better with long output code, where the padded data to entire code ratio minimized. Also successive redundant data give a better results on compressed LZW.

REFERENCES

[1] Havinal, R., G.V. Attimarad, and M. Giriprasad, EASR: Graph-based Framework for Energy Efficient Smart Routing in MANET using Availability Zones. *International Journal of Electrical and Computer Engineering (IJECE)*, 2015. Vol.5, No.6.

[2] Kumar, S.A., et al., An Empirical Critique of On-Demand Routing Protocols against Rushing Attack in MANET. *International Journal of Electrical and Computer Engineering (IJECE),, 2015*. 5(5).

[3] Costagliola, N., et al., Energy-and delay-efficient routing in mobile ad hoc networks. *Mobile Networks and Applications*, 2012. 17(2): p. 281-297.

[4] C. E. Perkins, E. M. Royer, and S. R. Das, "Ad Hoc On- Demand Distance Vector (AODV) Routing", Internet Draft, draft-ietf- manet- aodv-10.txt, work in progress, 2002.

[5] David B. Johnson and David A. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks", Computer Science Department, Carnegie Mellon University, Avenue Pittsburgh, PA 15213-3891.

[6] Kapang Lego, Pranav Kumar Singh, Dipankar Sutradhar, "Comparative Study of Adhoc Routing Protocol AODV, DSR and DSDV in Mobile Adhoc NETwork", *Indian Journal of Computer Science and Engineering* Vol. 1 No. 4 364-371, 2011.

[7] Hong, Y.S., A Control Packet Minimized Routing Protocol for Ad-hoc Wireless Networks. *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, 2014. Vol.12, No.2: p. 966-975.

[8] S. Singh, M. Woo, and C. S. Raghavendra, "Pamas: Power aware multi-access protocol with signalling for ad hoc network," *ACM Computer Communication Review*, pp. 5-26, July 1998.

[9] C. Schurgers, V. Tsiatsis, S. Ganeriwal, and M. Srivastava, "*Topology management for sensor networks: Exploiting latency and density*," in Proceedings of The Third ACM International Symposium on Mobile AdHoc Networking and Computing, 2002.

[10] W. Ye, J. Heidemann, and D. Estrin, "*An energy-efficient MAC protocol for wireless sensor networks*," in Proceedings of the The 21st AnnualJoint Conference of the IEEE Computer and Communications Societies, 2002.

[11] Jacobson, V. (1990). Compression TCP/IP for Low-Speed Serial Link, *RFC 1144*, 1990.

[12] C. Bormann, C. Burmeister, M. Degermark, H. Fukushima, H. Hannu, L.-E. Jonsson, R. Hakenberg, T. Koren, K. Le, Z. Liu, A. Martensson, A. Miyazaki, K. Svanbro, T. Wiebke, T. Yoshimura, H. Zheng, RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP and uncompressed, *Internet Engineering Task Force (IETF)*, Request for Comments: 3095, Category: Standards Track, 2001, pp. 1-168.

[13] Pelletier G., Sandlund K., RObust Header Compression Version 2 (ROHCv2): Profiles for RTP, UDP, IP, ESP and UDP-Lite, *Internet Engineering Task Force (IETF)*, Network Working Group, Category: Standards Track, Request for Comments: 5225, 2008, pp. 1-124.

[14] K. Sandlund, G. Pelletier, L.-E. Jonsson, The Robust Header Compression (ROHC) Framework, *Internet Engineering Task Force (IETF)*, Category: Standards Track, Request for Comments: 5795, 2010, pp. 1-41.

[15] G. Pelletier, K. Sandlund, L.-E. Jonsson, M. West, RObust Header Compression (ROHC): A Profile for TCP/IP (ROHC-TCP), *Internet Engineering Task Force (IETF)*, Category: Standards Track, Request for Comments: 6846, 2013, pp. 1-96.

[16] M. Majanen, P. Koskela, M. Valta, "*Constrained Application Protocol Profile for Robust Header Compression Framework*", in Proceedings of the Fifth International Conference on Smart Grids,Green Communications and IT Energy-awareTechnologies (ENERGY'15), Rome, Italy, 24-29 May 2015, pp. 47-53.

[17] Jeannot, E.; Knutsson, B. & Bjorkman, M. (2002). "*Adaptive Online Data Compression*", Proceedings of 11th IEEE International Symposium on High Performance Distributed Computing, pp. 379, ISBN 0-7695-1686-6, Edinburgh, Scotland, July 24-26, 2002.

[18] Krintz, C. & Sucu, S. (2006). "Adaptive On-The-Fly Compression", *IEEE Transaction on Parallel and Distributed Systems*, Vol.17, No. 1, January, 2006, pp. 15, ISSN 1045-9219.

[19] G. Montenegro, N. Kushalnagar, J. Hui, D. Culler, Transmission of IPv6 Packets over IEEE 802.15.4 Networks, *Internet Engineering Task Force (IETF)*,2007.

[20] R. Maulunida, A. Solichin, "Optimization of LZW Compression Algorithm With Modification of Dictionary Formation", *IJCCS (Indonesian J. Comput. Cybern. Syst.*, vol. 12, no. 1, pp. 73, 2018.

[21] WELCH, T. A. 1984." A technique for high-performance data compression". *IEEE Comput.* 17, 6, 8-19. 9

[22] ZIV, J. AND LEMPEL, A. 1978. "Compression of individual sequences via variable-rate coding". *IEEE Trans. Inform.* Theory 24, 5, 530–536.

[23] ZIV, J. AND LEMPEL, A. 1977. A "universal algorithm for sequential data compression". *IEEE Trans. Inform. Theory* 23, 3, 337–343.

[24] P. M. Nishad, R. M. Chezian, "Optimization of LZW (Lempel Ziv Welch) algorithm to reduce time Complexity for dictionary creation in encoding and decoding", *Asian Journal of Computer Science and Information Technology*, vol. 2, no. 5, 2012.

[25] Amit Setia and Priyanka Ahlawat, "*Enhanced LZW Algorithm with Less Compression Ratio*", Proceedings of ICAdC, AISC 174, Springer India 2013.