Indonesian Journal of Electrical Engineering and Computer Science
Vol. 19, No. 3, September 2020, pp. 1306~1315
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v19.i3.pp1306-1315

Increasing the efficiency of information transmission in communication channels

Bohdan Zhurakovskyi[1], Juliy Boiko[2], Vladymir Druzhynin[3], Irina Zeniv[4], Oleksander Eromenko[5]
[1,4]Department of Technical Cybernetics, National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute", Ukraine
[2]Department of Telecommunications and Radio Engineering, Khmelnytsky National University, Ukraine
[3]Department of Electrical Engineering and Radio Electronic Systems,
Taras Shevchenko National University of Kyiv, Ukraine
[5]Department of Physics and Electrical Engineering, Khmelnytsky National University, Ukraine

Article Info

Article history:

Received Feb 1, 2020
Revised Mar 18, 2020
Accepted Apr 9, 2020

Keywords:

Communication channels
Data compression
Encoding
Information compression
LZW method

ABSTRACT

This paper discusses compression methods focused on data transmission over communication channels. The characteristics of different algorithms for different types of incoming data are analyzed. The purpose of this study is to evaluate the speed of operation of each of the compression algorithms for different types of information and different compression parameters, on the basis of the obtained results to make recommendations for the application of compression methods in systems critical to the performance of the algorithm. Based on the results of the analysis, the methods of compression that can be used in communication channels are selected: LZW, LZH, Vitter and matrix. The practical research of the selected methods on different information flows (text, graphics, measurement data, combined data) was carried out, their comparative analysis was performed. Research has highlighted compression methods that give the most optimal results in each case. Comparative evaluation of algorithms for different parameters is made, the possibility of data compression implementation in systems running in real time is analyzed. Based on the results of the study, recommendations are made for the application of particular compression methods in specific conditions.

Copyright © 2020 Institute of Advanced Engineering and Science.
All rights reserved.

Corresponding Author:

Juliy Boiko,
Department of Telecommunications and Radio Engineering,
Khmelnytsky National University,
Instytuts'ka Street, 11, Khmelnytskyi, Khmelnytskyi Oblast, Ukraine.
Email: boiko_julius@ukr.net

1. INTRODUCTION

Improving the efficiency of information transfer in digital systems is very important in modern information flows, this is due to the unification of PCs in computer networks, and local computer networks into the global Internet. Recently, due to the extremely rapid increase in the amount of information transmitted, computer networksare overloaded due to the low bandwidth of existing communication channels. This problem can be solved in two ways: by replacing existing communication lines with new ones with greater bandwidth, or by introducing new methods of data compression [1-7]. The first method requires significant financial costs, in addition, it is not always possible to replace communication lines; more often it is more advisable to use existing lines for data transmission than to lay new ones. The second method allows you to significantly increase network performance, using existing communication channels, by reducing the amount of data transmitted over the network. This only requires replacing the software on the transmitting and receiving sides, which is much cheaper than laying new lines. Therefore, this solution to the problem is in most cases more preferable for control systems.

One of the important events in the history of information compression was the work of A. Lempel and J. Ziv. The algorithms LZ-77 and LZ-78 were subsequently improved and are the basis of most currently used data archivers [8]. The author [9, 10] proposes a compression algorithm that can be used to optimize another algorithm. This algorithm gives better compression ratio when inserted between move to front transform (MTF) and arithmetic coding (ARI). Because some files consist of hybrid contents, the ability to recognize contents regardless the file type, split it then compresses it separately with appropriate algorithm to the contents is potential for further research in the future to achieve better compression ratio. However, nothing is said about the speed of this algorithm. In article [11, 12], the authors propose a cascade compression method using one cascade of the RLE algorithm as the second one, an incremental algorithm. Compression efficiency increases by 50%, but how fast this algorithm works is unknown. Authors [13, 14] suggest data compression algorithm compared to other algorithm and enhance J-Bit encoding algorithm. This algorithm compress 37.51% data has been compressed and we use other decompress algorithm burrow wheel back ward and Front backward algorithm. Burrow wheel transform algorithm stands optimized used for the lossless figure’s compression in data warehouse. However, it is not clear how effective the proposed algorithm for compression and decompression rates is.

Paper [15-17] presents an efficient parallel approach to reduce execution time for compression Lempel–Ziv 77 (LZ77) and run length encoding with a K-precision (K-RLE) algorithms. The proposed OpenMP is an efficient tool for programming within parallel shared-memory environments. The improvement in compression ratio through an efficient parallel approach leads to reduction on transmission cost, reduction in storage space and bandwidth without additional hardware infrastructure. An overall performance evaluation shows arithmetic data compression algorithm with 46% which is better than LZ77 of 44% as well as K-RLE of 37% data compression algorithms. But the proposed algorithm is effective only in parallel environments with shared memory. The authors of [18-21] present a new approach to measuring performance and redundancy, which works on two coding methods, such as Huffman coding and Huffman coding with minimal dispersion, and get a better result than Huffman coding. But again, nothing is said about the compression rate.

In article [22, 23], the authors propose a new compression method and compare its performance with the popular adaptive Huffman coding. For comparison and analysis, the compression ratio, compression time, compression ratio and percentage of savings are used. The paper does not compare with other compression algorithms. In addition, an encryption algorithm (Caesar Cipher) has been added to ensure file protection and privacy, which slows down the proposed algorithm. As can be seen from the analysis of published articles, the question of the speed of the algorithms and compression methods, especially when transmitting data in real time, is not fully developed and is very relevant. When deciding on the use of a particular data compression algorithm in a particular information transfer system, it is necessary to evaluate the strengths and weaknesses of the algorithm, taking into account the conditions in which it will work. In addition to the compression coefficient, such parameters as speed, the required amount of memory for operation, the ability of the algorithm to adapt to the statistical properties of the incoming data, and when transmitting data in real time, the delay time of the transmitted data are important. Therefore, an algorithm that shows the best results in some conditions may work much worse or not at all for another system.

2. RESEARCH METHOD

In order to select the most suitable for information transfer in control systems from the whole set of compression methods, it is necessary to conduct a comparative assessment of the speed of the algorithm and the required memory resources for operation.

2.1. Key compression evaluations

To evaluate the effectiveness of the message compression procedure, several indicators of the degree of data compression are used. When evaluating the compression efficiency of text messages, the most widely used is the compression coefficient K_r, which characterizes the message volume V_r (in bits or bytes) at the compressor output after compression with respect to the initial volume V_u:

$$K_r = V_r / V_u \tag{1}$$

For the correct choice of the method of message compression during data transmission, one compression coefficient is not enough. The maximum transfer rate provided by each method must also be considered. The maximum compression rate is determined by the compressor speed, i.e., the number of characters N_c processed by the compressor per unit time T:

$$R_{\max} = N_c / T. \tag{2}$$

For some compression methods, the theory allows us to calculate the minimum achievable message volume J_{min} in a certain class of compression algorithms. Using such estimates, one can choose the method of presenting messages (compression method) selected in this class to characterize the efficiency coefficient.

$$\eta_R = J_{\min} / J. \tag{3}$$

The efficiency coefficient is some absolute characteristic of the chosen method of presenting messages in this class of signals.

Since the efficiency of algorithms varies greatly for different classes of signals, one of the tasks of researching each algorithm is the correct choice of the field of its application.

For a sufficiently accurate identification of the scope of each algorithm, a certain system of test signals should be installed. Signals of three main classes should be included in this system:

a) random;
b) analytical;
c) additive combinations of both

The amount of information contained in the input stream is equal to:

$$I_{EN} = \sum_{i=1}^{N_1} I(x_i), \tag{4}$$

By analogy with (4), the amount of information contained in the output stream, which consists of a set of N_2 elements, $N_2=\{y_1,y_2,...,y_{N2}\}$ of the alphabet q_2,

$$I_{EX} = \sum_{j=1}^{N_2} I(y_i), \tag{5}$$

To determine the compression ratio, it is necessary to obtain the values of the information transfer rates in the input and output streams. In this case, the information transfer rate in the input stream, bit/s,

$$R_{EN} = \frac{I_{EN}}{T_{EN}}, \tag{6}$$

where T_{EN} is the transmission time of the information array of the input stream, which, given the uniformity of information elements, is defined as:

$$T_{EN} = N_1 t_1, \tag{7}$$

where t_1 is the transmission duration of one element of the input information stream.

By analogy with (6) and taking into account (7) for the output stream, we can write:

$$R_{EX} = \frac{I_{EX}}{T_{EX}}, \tag{8}$$

where T_{EX} is the transmission time of the information array of the output stream.

2.2. Comparative compression techniques

One of the simplest compression methods is the series length coding method, also called longitudinal compression. In foreign sources, this method is called "Run Length encoding" (RLE encoding) [16]. This method is the easiest to implement, so it can be implemented both software and hardware. One of its main advantages is the high speed of compression and deployment, therefore, in those cases when the data from the source arrives at a high speed, it is the only possible, since other methods require large resources.

In addition, the method under consideration for its work does not require RAM at all, which facilitates its integration into existing control systems. However, the compression ratio [24] when using this method is quite low and does not completely eliminate the natural redundancy of the source.

The use of bit indicators. This method is also simple to implement, but has a lower speed compared to the method of encoding series lengths, since it requires the use of operations that work with individual bits. The method requires a small amount of memory to work. The compression ratio of the method strongly depends on the nature of the transmitted data: it is effective only when the data has a large number of spaces that are not grouped into continuous sections. In particular, in the absence of gaps, the volume of compressed text can be even larger than the original, which is often unacceptable for real-time information transfer systems. Compression of digital sequences. Like the previous ones, this method has high speed and low requirements for system resources. Its disadvantage is that when encoding a series of numbers, you need to know inadvance the size of theen code darea. This leads to a delay in the transmitted data andunevenoutput of compressed information by the encoder, which isundesirable for in formation transfer systems. Therefore, if the source generates only digital information, it is more advisable to initiallyen code it with four-bitcode combinations, or use thezone compression method [25].

The matrix compression method refers to two-dimensional methods. It is based on the search for duplicate data chains. The method works as follows: matrices of repeating sequences of characters are searched in the flow of incoming information. Before starting the transmission of the data block, information about the matrices found is transmitted, each of which is assigned a specific code. Then the information itself is transmitted, in which each matrix found is encoded with only one line, while the dimensions of each matrix are set in the additional information block [26, 27]. This compression method works well for information that has a clear structure, for example, for database records. If one or more database fields have the same value for several consecutive records, the matrix method will effectively compress such data. The method also compresses graphical information well when the image has large monochrome areas.

Diatomic coding. In the vocabulary constructions of various languages, there are always stable letter combinations (strings) that are more common than other characters, and the percentage of combinations of two or three letters is much higher than longer combinations. This property is used in diatonic coding, in which compression occurs by replacing a pair of characters with special code words.

The maximum number of pairs to be replaced is determined by the length of the output words. The total number of the most common pairs of characters is relatively small. As the studies of a number of authors have shown [28, 29], in the English text the 25 most frequently encountered pairs make up about 34% of the total number of characters in the text. This compression method has high speed when implementing a list of pairs of characters using the tabular method, however, in this case it requires a significant amount of memory. The method is suitable for compressing textual information in a certain language, however, in the case of binary data transmission, when the statistical dependence between adjacent characters is weak or absent, this compression method will not have any effect.

Coding bynon-uniform codes (Huffman). This class of methods is one of the best known. Althoughin som ecases it provide san acceptable compression ratio, there dundancy of the source when using uneven coding is not completely eliminated, since this method doesnot take into account the interdependence between individual characters. Inaddition, ithas a lowerspeed, since it works within dividual bits. The compression ratio can bein creased by combining characters in groups, however, this greatly increases the size of the code tree. The arithmetic coding algorithm is more complicated to implement and has a fairly low speed, since at least one multiplication operation is required for each byte of the encoded message. When using this method, the number of bits spent on coding a message is closest to the source entropy. However, this method also does not take into account the interdependence of characters. Therefore, in data transmission systems, it is more expedient to use simpler compression methods [3].

Vitter coding. This method uses a tree data structure called a "floating tree" because the pointers of the parent and its child nodes are supported implicitly. Each block has only pointers of the parent and right child nodes of the block leader. Due to the continuity of the memory where external and internal nodes are stored, the position of the parent and child nodes of the tree of other block nodes can be determined for a fixed time by calculating the offset from the pointers of the parent and its right child node of the block leader. This allows the node to slide around the block, modifying a constant number of pointers. Thus, the procedure takes a constant time, which allows real-time encoding and decoding. This property is very important when using compression in synchronous data transmission systems [30].

Encoding method LZW, LZH. These methods provide the best compression ratios, which leads to their use in most data archivers. LZW data compression algorithms are fairly simple. To reduce the search time for a string in the code table, it is achieved by using the method of direct access to the table, which is called "hashing". Due to the use of "hashing" which increases the speed of searching a row in a table and a larger compression ratio, the LZW algorithm is widely used in modern archivers such as ARJ, PKZIP.

The LZH algorithm uses a statistical and vocabulary compression method, which allows two compression methods to be used during archiving, which gives a significant compression ratio. A similar algorithm is used in the V.42bis protocol [31]. Thus, based on the analysis of the presented compression methods and consideration of their strengths and weaknesses, the following were distinguished: Vittera compression; LZH compression; LZW compression. The aim of the research is the practical analysis of each of the methods of data compression for the different nature of the data being compressed and the development of recommendations on the use of compression methods in transmitting data in communication channels. As the main indicators of compression efficiency during research, the compression speed and data delay at the encoder output were selected. Measurements are made separately for each of the parameters, while the dependence of performance indicators on the size of the transmitted data block is investigated.

The following information fragments were used as input for the encoder:

a) information in text form;
b) graphic information;
c) measuring information from sensors;
d) mixed data that contains graphical information, tables, text formulas

For the convenience of encoding, the alphabet of the input and output code is chosen equal to 256 characters (one byte is used to encode one character). In cases where the output code requires the expansion of the alphabet, an alphabet of 128 characters is used at the input (the lower 7 bits of the code combination are used). The measurement technique is as follows. The Test Sequence Generator produces a data snippet of one of four data types. This fragment enters the processing and display module, which calls the specified compression procedures by various methods and receives compressed sequences from them. When measuring the compression speed, it also starts the timer. Comparing the timer readings before the call and after calling the compression procedure, it obtains a compression time. The measurement results are then fed to a data display module that represents them as graphs.

3. RESULTS AND ANALYSIS

The purpose of this study is to evaluate the speed of each of the compression algorithms for various types of information and various compression parameters, based on the results obtained, to develop recommendations on the use of compression methods in systems critical to the performance of the algorithm. Testing was conducted on processors, Intel Pentium - 2.4. To exclude the influence of the size and speed of the second level cache on the speed of the algorithm, the system cache was turned off. For testing, we used data fragments with a length of 4096 bytes, while the measurement was performed in a cycle of 1000 times to improve the accuracy of the results.

3.1. Performance and compression time by LZH method

Figure 1 shows the dependence of the compression time on the size of the compressible block Table 1. As can be seen from the figure, the larger the compressible block, the more time is spent on its compression.

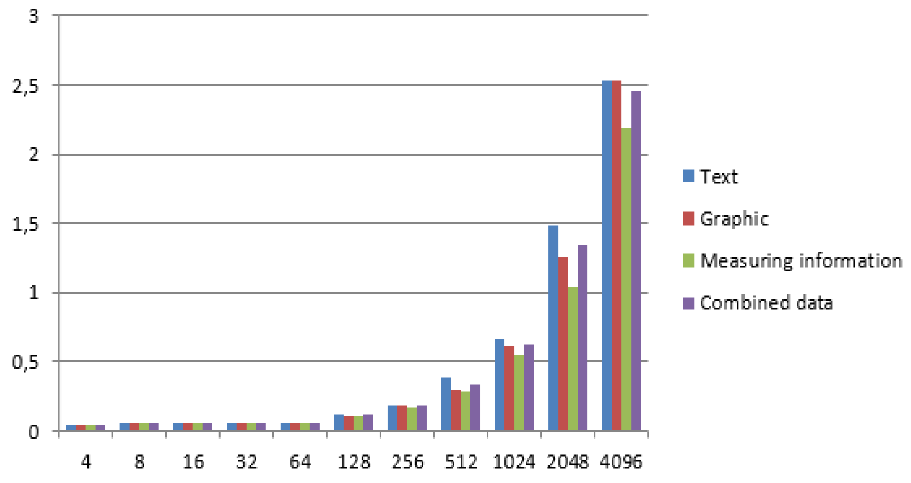

Figure 1. Compression time versus size of the compressible block for the LZH algorithm

Table 1. Dependence of compression time on block size for the LZH algorithm

	Type of Information	Block size										
		4	8	16	32	64	128	256	512	1024	2048	4096
L Z H	Text	0.051	0.06	0.06	0.06	0.06	0.115	0.180	0.38	0.66	1.48	2.53
	Grafic	0.051	0.06	0.06	0.06	0.06	0.112	0.182	0.30	0.61	1.26	2.53
	Measuring information	0.051	0.06	0.06	0.06	0.06	0.11	0.170	0.28	0.55	1.04	2.19
	Combined data	0.051	0.06	0.06	0.06	0.06	0.115	0.180	0.38	0.66	1.48	2.53

For measuring information, the compression time is the smallest, this is due to the same type of information, for which the code table is not very large, which reduces the search time for the corresponding code. For the rest of the data, the time increases with an increase in the compressible block. Let us estimate the maximum throughput of the algorithm for the Pentium processor - 2.4. An array of 4096 characters is encoded 1000 times per T_{max} = 27.45 s, therefore, the processing time of one character is t_c =27.45/(4096·1000)=6.70 μs. Hence, the throughput of the algorithm is R_{max} = 147656.73 characters/s.

3.2. Speed and compression time by LZW method

Figure 2 shows the dependence of the compression time for the LZW method on the block size Table 2.

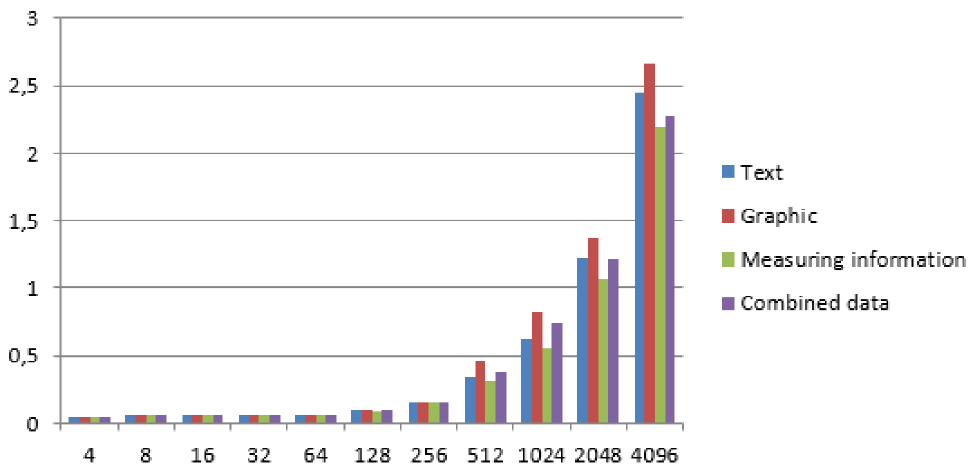

Figure 2. Compression time versus size of the compressible block for the LZW algorithm

Table 2. Dependence of compression time on block size for the LZW algorithm

	Type of Information	Block size										
		4	8	16	32	64	128	256	512	1024	2048	4096
L Z W	Text	0.051	0.06	0.06	0.06	0.06	0.10	0.155	0.34	0.62	1.23	2.45
	Grafic	0.051	0.06	0.06	0.06	0.06	0.10	0.160	0.46	0.82	1.38	2.66
	Measuring information	0.051	0.06	0.06	0.06	0.06	0.095	0.150	0.31	0.56	1.07	2.19
	Combined data	0.051	0.06	0.06	0.06	0.06	0.10	0.160	0.46	0.82	1.38	2.45

As can be seen from the figure, the least compression time has measurement information. Compressing a block of size 1024 bytes still gives the allowable time required for compression in communication channels. Over 1024 bytes, the time of the compressed information is increased to seconds, which leads to a delay and which is unacceptable in real-time systems. Let us estimate the maximum throughput of the algorithm for the Pentium processor - 2.4. An array of 4096 characters is encoded 1000 times per T_{max} = 23.57s, therefore, the processing time of one character is t_c =27.45/(4096 · 1000)=6.72 μs. Hence, the throughput of the algorithm is R_{max} = 147656.73 characters/s.

3.3. Performance and compression time by Vitter

Figure 3 shows the dependence of the compression time for the Vitter method on the block size Table 3.

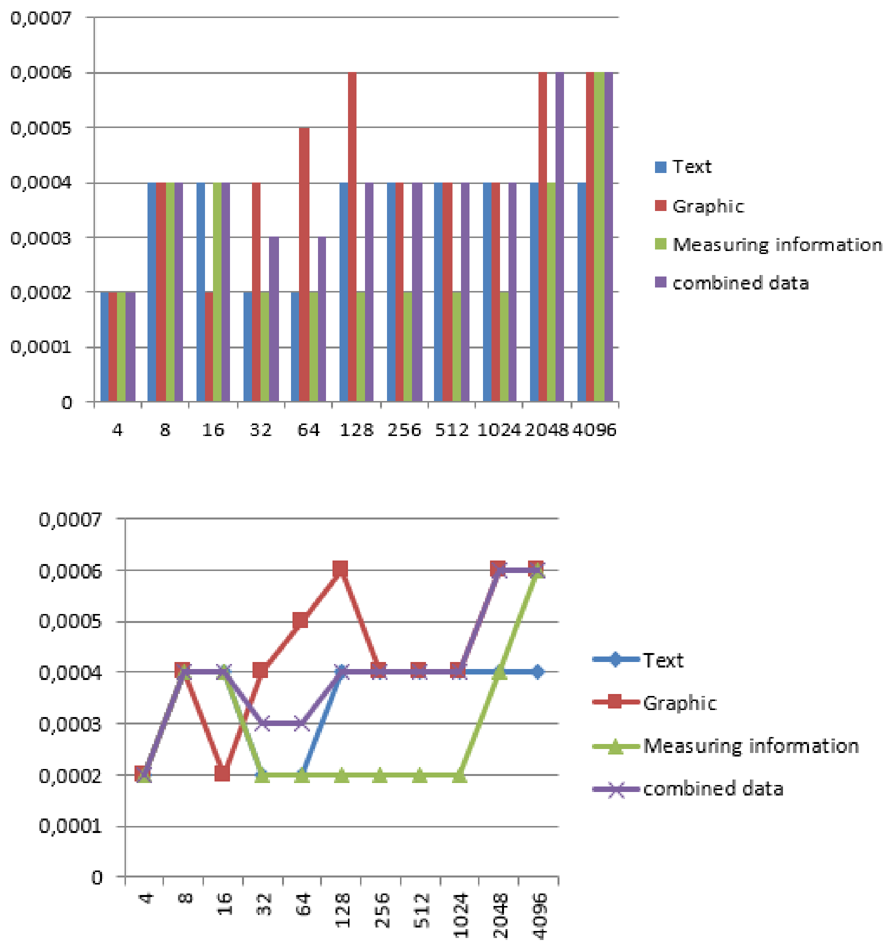

Figure 3. Compression time versus size of the compressible block for the Vitter algorithm

Table 3. Dependence of compression time on block size for the vitter algorithm

	Type of Information	4	8	16	32	64	128	256	512	1024	2048	4096
Vitter	Text	0.0002	0.0004	0.0004	0.0002	0.0002	0.0004	0.0004	0.0004	0.0004	0.0004	0.0004
	Grafic	0.0002	0.0004	0.0002	0.0004	0.0005	0.0006	0.0004	0.0004	0.0004	0.0006	0.0006
	Measuring information	0.0002	0.0004	0.0004	0.0002	0.0002	0.0002	0.0002	0.0002	0.0002	0.0004	0.0006
	Combined data	0.0002	0.0004	0.0004	0.0002	0.0002	0.0004	0.0004	0.0004	0.0004	0.0004	0.0004

This compression method gives the smallest compression time from the considered algorithms, as can be seen from the figure. The minimum compression time for the measurement information is explained by the fact that the procedure that constructs the tree is performed in a constant time, regardless of the size of the compressible block. Worst of all, graphic information is compressed. For textual and combined information, it can be seen that compression occurs in almost the same time, and some jumps are explained by the influence of external influences, although the time was estimated in a cycle of 1000 times. Let's estimate the maximum throughput of the algorithm for the Pentium processor - 2.4. An array of 4096 characters is encoded 1000 times per T_{max} =0.036 s, therefore, the processing time of one character is t_c=0.036/(4096 · 1000)=0.000117 µs. Hence the throughput of the algorithm R_{max} = 8.5449 · 109 characters/s.

3.4. Performance and time compression matrix method

Figure 4 shows the dependence of the compression time for the matrix method on the block size Table 4.

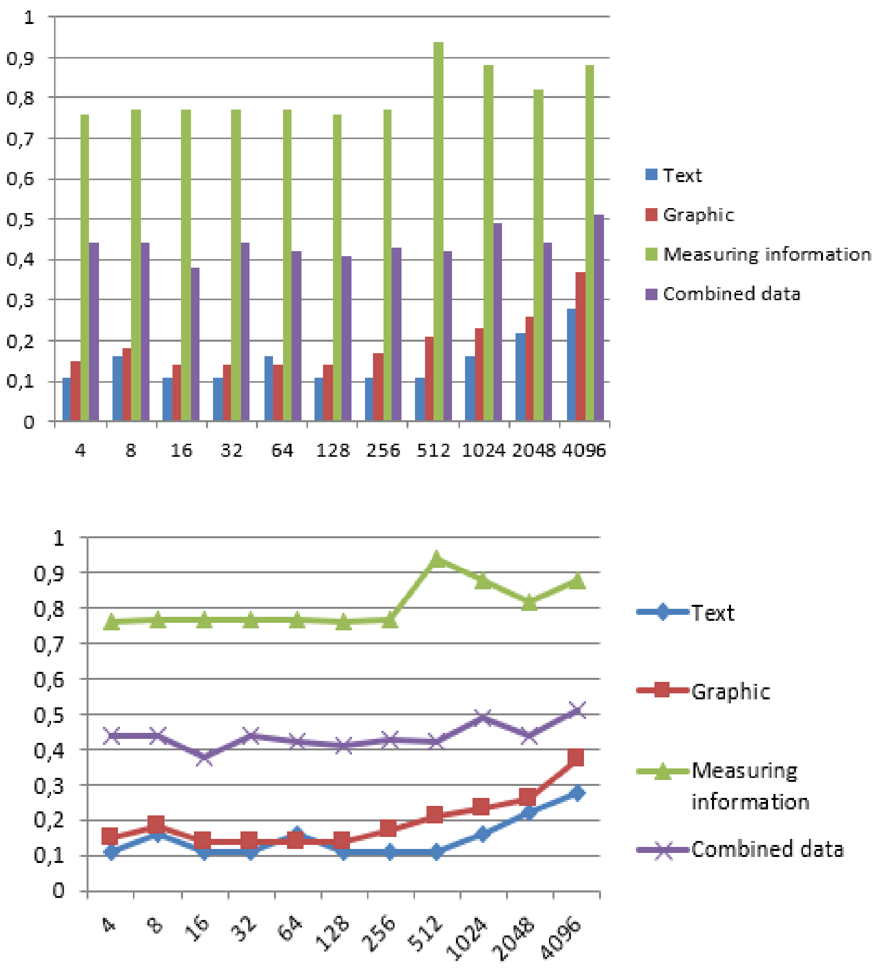

Figure 4. Compression time versus size of the compressible block for the matrix algorithm

Table 4. Dependence of compression time on block size for the matrix algorithm

Type of Information		Block size										
		4	8	16	32	64	128	256	512	1024	2048	4096
Matrix method	Text	0.76	0.77	0.77	0.77	0.77	0.76	0.77	0.94	0.88	0.82	0.88
	Grafic	0.44	0.44	0.38	0.44	0.44	0.44	0.49	0.44	0.50	0.49	0.61
	Measuring information	0.76	0.77	0.77	0.77	0.77	0.76	0.77	0.94	0.88	0.82	0.88
	Combined data	0.11	0.16	0.11	0.11	0.16	0.11	0.11	0.11	0.16	0.22	0.28

The matrix method, as can be seen from the figure, performs compression almost the same time, but for different flows in different ways. For measuring information, the time is greatest, this is due to the fact that the search and adaptation to this type of information takes the longest time. For text, the time is less than in the previous case, this is due to the fact that compression does not occur and uncompressed information is output. Also, this dependence is observed for combined information. Let's estimate the maximum through put of th ealgorithm for the Pentium processor - 2.4. Anarray of 4096 characters is encoded 1000 time sin T_{max} =4.2 s, therefore, the processing time of one character t_c =4.2/(4096 · 1000)=0.000001024 s. Hence, the through put of the algorithm R_{max}=975238.74 characters/s.

4. CONCLUSION

The measurement of the speed of the algorithms. The dependence of the compression rate on the size of the data block and the entropy of the information flow on various types of processors was investigated. Based on the results of measurements, the throughput of each compression algorithm for each of the processors was calculated. The obtained values allow, based on the speed of information transfer in the communication channel, to select the most optimal hardware platform for building a compressor. For systems in which information is transmitted in text form, a significant reduction in message redundancy can be obtained using the LZH, LZW methods. These methods are text-oriented and, when compressed, give a large coefficient and good compression time. For both methods, the least compression time has measurement information. This is due to the same type of information, for which the code table is not very large, which reduces the search time for the corresponding code. For the rest of the data, graphic information, text and combination, the time increases as the compressible block increases. The Vitter method gives the least compression time from the considered algorithms. The minimum compression time of the measurement information is explained by the fact that the procedure that produces the construction of a tree is performed over a constant time regardless of the size of the compressible block. Most of the time is devoted to the compression of graphic information. Compression time for textual and combined information occurs at virtually the same time. The matrix method performs compression almost at the same time, but for different threads differently. For measurement information, the time is greatest, this is explained by the fact that the search and adaptation to this species takes the greatest time.

REFERENCES

[1] B. Zhurakovskiy and N. Tsopa, "Assessment technique and selection of interconnecting line of information networks," *2019 3rd IEEE International Conference on Advanced Information and Communications Technologies (AICT)*, Lviv, pp. 71-75, 2019.

[2] N. Asilah Khairi, *et al.*, "Performance evaluation of arithmetic coding data compression for internet of things applications," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 13, no 2, pp. 591-597, Feb 2019.

[3] H. Noori Saad, *et al.*, "A new compression technique in MANET: compressed-LZW algorithm," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 16, no. 2, pp. 890-896, Nov 2019.

[4] M. Safieh and J. Freudenberger, "Efficient VLSI architecture for the parallel dictionary LZW data compression algorithm," in *IET Circuits, Devices & Systems*, vol. 13, no. 5, pp. 576-583, 8 2019.

[5] S. Aruna Deepthi, *et al.*, "RTL Implementation of image compression techniques in WSN," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 3, pp. 1750-1756, Jun 2019.

[6] Y. Wang, *et al.*, "Compression algorithm of road traffic data in time series based on temporal correlation," in *IET Intelligent Transport Systems*, vol. 12, no. 3, pp. 177-185, 4 2018.

[7] J. Boiko, I. Kovtun and S. Petrashchuk, "Productivity of telecommunication systems with modified signal-code constructions," *2017 4th IEEE International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T)*, Kharkov, pp. 173-178, 2017.

[8] J. Ziv and A. Lempel, "Compression of individual sequences via variable-rate coding," *in IEEE Transactions on Information Theory*, vol. 24, no. 5, pp. 530-536, Sep 1978.

[9] I Made Agus Dwi Suarjaya, "A new algorithm for data compression optimization," *International Journal of Advanced Computer Science and Applications*, vol. 3, no. 8, pp. 14-17, 2012.

[10] O. F. Abdel Wahab, *et al.*, "Hiding data in images using steganography techniques with compression algorithms," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 17, no. 3, pp. 1168-1175, Jun 2019.

[11] A. Singh and Y. Bhatnagar, "Enhancement of data compression using incremental encoding," *International Journal of Scientific & Engineering Research*, vol. 3, no. 5, pp. 1-5, May 2012.

[12] D. S. Bhadane and S. Y. Kanawade, "Comparative study of RLE & K-RLE compression and decompression in WSN," *2016 3rd International Conference on Advanced Computing and Communication Systems (ICACCS)*, Coimbatore, pp. 1-5, 2016.

[13] S. Ambadekar, *et al.*, "Advanced data compression using J-bit algorithm," *International Journal of Science and Research*, vol. 4, no. 3, pp. 1366-1369, Mar 2015.

[14] U. Nandi and J. K. Mandal, "A Compression Technique Based on Optimality of LZW Code (OLZW)," *2012 IEEE Third International Conference on Computer and Communication Technology*, Allahabad, pp. 166-170, 2012.

[15] F. S. Mahammad and V. M. Viswanatham, "Performance analysis of data compression algorithms for heterogeneous architecture through parallel approach," *Journal of Supercomputing*, vol. 76, pp. 2275-2288, 2018.

[16] D. Salomon and G. Motta, "Handbook of data compression," Publisher: Springer-Verlag London Limited. 2010.

[17] T. Nishimoto and Y. Tabei, "LZRR: LZ77 parsing with right reference," *2019 IEEE Data Compression Conference (DCC)*, Snowbird, UT, USA, pp. 211-220, 2019.

[18] G. S. Sandeep, B. S. S. Kumar and D. J. Deepak, "An efficient lossless compression using double Huffman minimum variance encoding technique," *2015 International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)*, Davangere, pp. 534-537, 2015.

[19] T. Koya Poolakkachalil and S. Chandran, "Summative stereoscopic image compression using arithmetic coding," *Indonesian Journal of Electrical Engineering and Informatics,* vol. 7, no. 3, pp. 564-576, Sep 2019.

[20] C. Atika Sari, G. Ardiansyah, D. R. I. M. Setiadi, E. H. Rachmawanto, "An improved security and message capacity using AES and Huffman coding on image steganography," *TELKOMNIKA (Telecommunication Computing Electronics and Control),* vol. 17, no. 5, pp. 2400-2409, Oct 2019.

[21] J. Boiko, V Tolubko, O Barabash, O Eromenko, Y Havrylko, "Signal processing with frequency and phase shift keying modulation in telecommunications," *TELKOMNIKA (Telecommunication Computing Electronics and Control),* vol. 17, no. 4, pp. 2025-2038, Aug 2019.

[22] K. Sharma and K. Gupta, "Lossless data compression techniques and their performance," *2017 International Conference on Computing, Communication and Automation (ICCCA),* Greater Noida, pp. 256-261, 2017.

[23] W. Wang and W. Zhang, "Huffman Coding-Based Adaptive Spatial Modulation," *in IEEE Transactions on Wireless Communications*, vol. 16, no. 8, pp. 5090-5101, Aug. 2017.

[24] A. Yazdanpanah and M. R. Hashemi, "A new compression ratio prediction algorithm for hardware implementations of LZW data compression," *2010 IEEE 15th CSI International Symposium on Computer Architecture and Digital Systems*, Tehran, pp. 155-156, 2010.

[25] P. Gillman, "Data handling and text compression," *Journal of Information Science*, vol. 18, no. 2, pp. 105-110, Feb 1992.

[26] J. D. Kotulski, "The parallel implementation and accuracy of matrix compression in the method of moments code EIGER," *2018 IEEE International Applied Computational Electromagnetics Society Symposium (ACES)*, Denver, CO, pp. 1-2, 2018.

[27] J. Ding, Y. Huang, P. Lin, S. Pei, H. Chen and Y. Wang, "Two-Dimensional Orthogonal DCT Expansion in Trapezoid and Triangular Blocks and Modified JPEG Image Compression," in *IEEE Transactions on Image Processing*, vol. 22, no. 9, pp. 3664-3675, Sept. 2013.

[28] K. Kim, C. Lee and H. Lee, "A Sub-pixel gradient compression algorithm for text image display on a smart device," *in IEEE Transactions on Consumer Electronics*, vol. 64, no. 2, pp. 231-239, May 2018.

[29] T. C. Bell, I. H. Witten and J. G. Cleary. Text compression, Englewood Cliffs. Publisher: Prentice-Hall, 1990.

[30] M. W. Maier, "Algorithm evaluation for synchronous data compression," *Proceedings DCC '95 Data Compression Conference*, Snowbird, UT, USA, pp. 444, 1995.

[31] J. Ziv and Y. Hershkovitz, "Another look at universal data compression," *Proceedings of 1994 IEEE International Symposium on Information Theory*, Trondheim, Norway, pp. 11, 1994.