

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/337390102>

A Byte Pattern Based Method for File Compression

Chapter · November 2019

DOI: 10.1007/978-3-030-34989-9_10

CITATION

1

READS

226

5 authors, including:



Jose Luis Hernández Hernández

TecNM / Instituto Tecnológico de Chilpancingo

52 PUBLICATIONS 381 CITATIONS

[SEE PROFILE](#)



Mario Hernández Hernández

Universidad Autónoma de Guerrero

26 PUBLICATIONS 130 CITATIONS

[SEE PROFILE](#)



Sajad Sabzi

Sharif University of Technology

62 PUBLICATIONS 704 CITATIONS

[SEE PROFILE](#)



Alejandro Fuentes-Penna

El Colegio de Morelos

104 PUBLICATIONS 171 CITATIONS

[SEE PROFILE](#)

Some of the authors of this publication are also working on these related projects:



A Neural Network for Forecasting [View project](#)



Air, Water and Waste Management [View project](#)

Título: A Byte Pattern Based Method for File Compression

Autores: José Luis Hernández-Hernández

Mario Hernández-Hernández

Sajad Sabzi

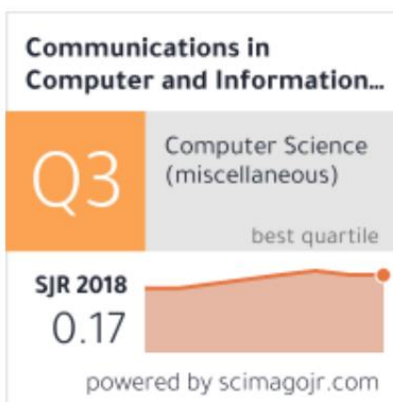
Mario Andrés Paredes-Valverde

Alejandro Fuentes Penna

Revista: Communications in Computer and Information Science book series (CCIS, volume 1124), pp. 122–134, 2019.
Springer Nature Switzerland AG 2019.

Factor de Clasificación: Q3 en las categorías de SJR (2018)

Impacto de acuerdo a: SCImago Journal & Country Rank








Aceptado: Agosto-2019

Publicado: Noviembre-2019

Digital Object Identifier: https://doi.org/10.1007/978-3-030-34989-9_10



A Byte Pattern Based Method for File Compression

José Luis Hernández-Hernández¹(✉) , Mario Hernández-Hernández² ,
Sajad Sabzi³ , Mario Andrés Paredes-Valverde⁴ ,
and Alejandro Fuentes Penna⁵ 

¹ TecNM/Technological Institute of Chilpancingo, Chilpancingo, Mexico
`joseluis.hernandez@itchilpancingo.edu.mx`

² Autonomous University of Guerrero, Chilpancingo, Mexico
`mhernandezh@uagro.mx`

³ University of Mohaghegh Ardabili, Ardabil, Iran
`sajadsabzi2@gmail.com`

⁴ University of Murcia, Murcia, Spain
`marioandres.paredes@um.es`

⁵ TecNM/CIIDET, Querétaro, Mexico
`afuentes@ciidet.edu.mx`

Abstract. This research presents a method to allows the data compression from a file containing any type of information by combining the pattern theory with the theory of data compression. This proposal can reduce the storage space of a file data from any kind of computer, platform or operating system installed on that computer. According to the fundamentals of patterns, a pattern is a regularity of bytes contained within a file with self-similarity characteristics; if this concept applies to data files, we find certain amounts of auto-similar or patterns repeated several times throughout the file; with a store data representation and being referenced, at a certain point data can be recovered from the original file without losing a single data, and consequently saving space on the hard disk.

In the search for various ways to compress data, led me to analyze and implement the proposed methodology in a beta mode compression software for Windows 10, which presents very compromising results.

Keywords: Patterns · Data compression · Tiles · Mathematical pattern

1 Introduction

This research proposes a file compression method that allows reducing file size so that it takes up less space on the computer's hard drive. This method does not affect the content or structure of the file, it simply reduces the space it occupies [9].

In computer science, the purpose of compressing a data file is to use an algorithm and apply it to the data so that it has a transformation and takes

up less storage space. Depending on the data types contained in the file, the implemented algorithm may be more or less effective. Aiming to perform the file compressing process, a well-crafted algorithm, a large memory capacity, and a processor with good processing speed are needed [14,17].

Data compression typically applies when a file needs to be sent over the Internet because email and messaging applications put limits on the size of the files that can be sent. Therefore, the file is usually compressed to be sent over the network. Data compression, which involves transforming data into a given format, helps to reduce storage and communication costs [8,11,12,21].

The file compression method proposed in this work uses the fundamentals of patterns approach for data compression purposes. A pattern in an image is the minimum unit of the image that bears a resemblance in colour and size and that is repeated several times in an image or group of images [23].

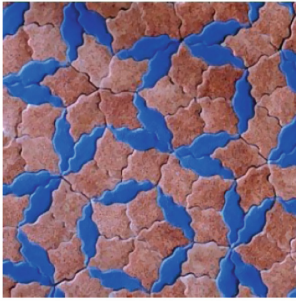
The most basic patterns of images are commonly called tiles based on repetition and recurrence. A single template, tile or cell, is combined by unchanged duplicates or modifications [1].

The fractals are another kind of pattern, that are geometric objects whose basic structure, fragmented or seemingly irregular, is repeated at different scales [20]. The term was proposed by mathematician Bernot Mandelbrot in 1975 and derives from Latin fractus, meaning broken or fractured. Many natural structures are fractal type as it says [3]. The key mathematical property of a genuinely fractal object is that its fractal metric dimension is a rational number and is not an integer [4].

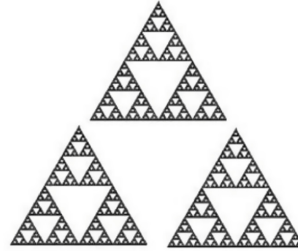
The type of pattern that can be used in flat files are adjacent bytes, which form syllables, word fragments, words, or similar fragments [15]. In a data file that contains any type of information (image, video, text, etc.), the bytes that are stored one after the other correspond to the ASCII code. This group of elements can be letters, numbers, special characters, or characters of control; so that each byte has a value between the range 0 to 255 according to ASCII code. In this way, a block of bytes can be simplified and represented by a single integer [19].

The form of self-similarity representation has much to do with the fractal concept established by Bernot Mendelbrot. This property allows to recognize elements that have the characteristics of the whole, from which it is extracted and it allows that to replicate that whole can be regenerated. Figure 1 shows the 4 types of patterns described above.

Data compression is useful because it helps to reduce the use of expensive resources, such as disk space or the bandwidth to transmit data [7]. On the negative side, compressed data must be decompressed to be seen and this additional process may be detrimental to some applications. For instance, a compression scheme for video may require expensive hardware, so that the video is decompressed fast enough as to be seen while decompressing (there is the option to decompress the video completely before seeing it, but this is inconvenient and storage space required to see the decompressed video). Thus, the design of data compression schemes implies compensation among several factors such as



(a) Tile



(b) Fractal

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

(c) Mathematical pattern

According to the fundamentals of patterns, it is said that a pattern is a regularity of bytes found within a file and which has the characteristic of self-similarity.

Patterns: the, of, pattern, is.

(d) Byte pattern

Fig. 1. Diversity of patterns that can be used.

number of bytes in the file, degree of compression, the compression algorithm used and necessary computing resources to compress/decompress the data [2, 18].

Nowadays, a file compressor application is installed on many computers. This application allows grouping multiple files and/or folders into a single package to reduce the space you occupy on the storage medium.

The two most used file compressors are: WinZip © and WinRAR ©. The most important characteristics of these tools are described below:

- WinZip © is a commercial file compressor developed by WinZip Computing (formerly known as Nico Mak Computing) that works on Microsoft Windows. It can handle several additional file formats. (WinZip © is a registered trademark of Microsoft Corporation ®) [22].
- WinRAR © is a useful application that allows creating, managing and controlling directories. It is the most widely used solution for decompression and compression of information, helping to reduce the size and improve response time when sending content or creating a backup. It is compatible with any type of document or application and provides a RAR and UnRAR file decompressor for multiple platforms. (WinRAR © is a registered trademark of RAR-BAL ®) [13].

WinZip © and WinRAR ©, compress any type of data contained in a file and can subsequently decompress it without data loss. It performs data compression

using byte patterns and strings that repeat throughout the file and perform exactly the same function.

2 Materials and Methods

To perform data compression, is considered of the fact that groups of elements exist that have the basic properties of self-similarity and through which it is possible to reproduce all and each of the combinations that can be found in ASCII code.

The first step of the file compression process consists in representing 2 adjacent bytes as a pattern of its original elements. The result of this task is a representation of each pair of adjacent bytes of a defined block of bytes. This process is repeated with the new patterns found in order to create patterns of 2, 4, 8, 16, 32, . . . , n byte groups.

Thanks to this process, the data storage is very efficient since instead of storing the entire block of bytes, only its pattern-based representation is stored.

The file compression and decompression processes require a master pattern file that has the format shown in Table 1.

Table 1. Data structure of the master pattern file.

Pattern number	Left byte	Right byte
Any integer from 256	Any integer value with its respective sign	Any integer value with its respective sign

Where:

- a. Pattern number. It is a positive integer that is generated with values ranging from 0 to 255 because they are taken directly from the ASCII code to represent each of the respective characters (some not printable and other control).
- b. Left byte. It is a positive or negative number, which corresponds to the first byte of the pair of bytes that you want to handle as a pattern.
- c. Right byte. It is a positive or negative number, which corresponds to the second byte of the pair of bytes that you want to handle as a pattern.

The data structure above presented corresponds to each pattern stored in a file known as Pattern Master File. Once file compression using integer patterns is performed, a file composed of the next two parts is obtained:

- a. Header. It contains data that identifies it as a patterns file, file size, as well as the name of the original file (original unit, path, file name, and your extension), as shown in Table 2.
- b. Data area. It contains only integer numbers that correspond to patterns consisting of a group of three integer values.

Table 2. Structure of the pattern file header.

Identifier	Size in bytes	File name
This field is composed of 2 bytes as follows: ASCII code: 80 letter ‘P’; ASCII code: 97 letter ‘a’	This field stores in an integer value, the amount in bytes occupied by the file name	This field contains: storage unit, path, file name and your extension (original)

2.1 Compressor Architecture

Regarding the man-machine interface, the project includes tools that allow users to build the form look & feel of user interfaces [16], to configure the screen layout. These features allow users to configure the information displayed on the screen in terms of information density and attributes of deployment such as colours, types of lines, filling patterns of the figures, sizes, titles, resolution, etc. Figure 2 depicts the architecture of the file compression application proposed in this work.

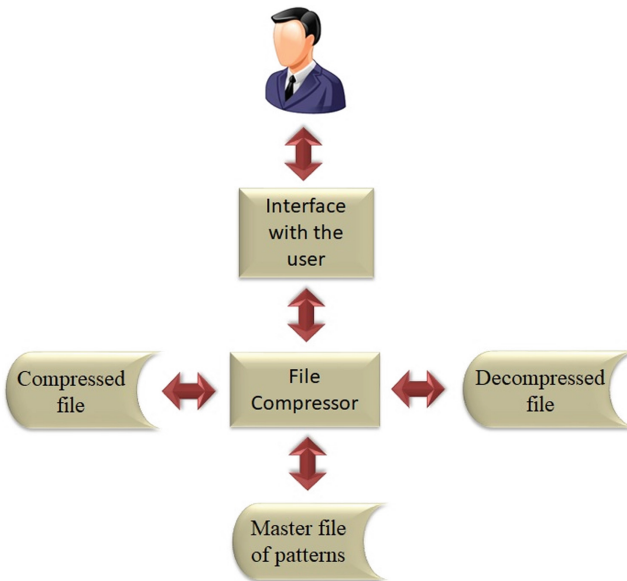


Fig. 2. Basic scheme of the file compressor.

It also uses context-sensitive support to assist the user in the operation of the interface and of interaction techniques and dialogues. The reliability and acceptance of the system involve some considerations that must be taken into account in all aspects of design and development to minimise the possibility of data failures.

2.2 Compression Process

For developing the file compression application here proposed, it is required to analyse all the file to be compressed and search for repetitive patterns which will be considered as similar auto entities. Then, these patterns are represented as a unique integer numeric value and are stored in a pattern file that will be used by the decompression process. This file is also known as data dictionary.

The new patterns that are generated are stored in a file, however, any known pattern is no longer stored; it is simply used for compression and the decompression of a file.

At the time of finding new patterns is always verified the pattern obtained, against those that already exist and if it exists takes its entire numerical value to represent it in the file being compressed.

Unfortunately, the patterns are generated in a static form and are defined as the data necessary to represent the pattern [5,6,10,17]. These data are represented by the structure shown in Table 3.

Table 3. Structure to store the patterns that are obtained.

Pattern	Left element	Right element
One number	One number	One number

The structure of Table 3 will be used and stored in a file called MASTER.CFP (Compressed file with patterns) and will be found in the unit C in the folder C:\WINDOWS of the computer in use and with this form of compression, stored patterns are simply used. A drawback is that each time a new pattern is generated, it is stored in the master file of patterns that will continue growing. The Algorithm 1 used to compress the data of a file using patterns is shown below.

A training process consisting of testing about 100 files with different content is performed. This process aims to store the most amount of patterns that may be required for the decompression operation in the master file of fractals.

When installing the software, it will copy the master file of fractals to the root level on the computer in use.

When performing the compression process, the master file of fractals will grow a little and at that time the new patterns will be updated in a copy that will be in the cloud. Such a copy of the cloud updates the fractal master file in real-time installed on any computer.

2.3 Decompression Process

The decompression process starts by opening a file that has been compressed with the compression project software. In the file opening dialog box, the application only shows those files that have extension CFP.

Algorithm 1. Algorithm to compress a file using patterns.

```

1: Open file to compress (source)
2: Open file where the compressed data will be (output)
3: while (not end of file) do
4:   Read 50 bytes from the source file
5:   Take numerical value according to the ASCII code of each character
6:   Take pairs of bytes to create patterns
7:   Check if the pattern exists in the pattern master
8:   If the pattern does not exist, store it in the master pattern file
9:   Rebuild pattern repeatedly
10:  Write the pattern to the output file
11: end while
12: Close files

```

Once a file with the specified extension is opened, the application verifies that the characters “P” and “a” exist in the first 2 bytes. If the file does not have this identifier, it means that the file is damaged and/or was not created with this application.

In this process, a string of characters containing information such as unit, the path, file name, and the extension with which the file will be created is used to determine where it will be decompressed. Then, integer values or patterns are read, one by one until you reach the end of the file to decompress.

The decompression process is performed for each integer value or pattern identified. The Algorithm 2 used to decompress data from a file using patterns is shown below.

Algorithm 2. Algorithm to decompress a file using patterns.

```

1: Open file to decompress
2: Open the output file
3: Read the header of the file to be decompressed
4: Verify that the first 2 bytes have the characters “Pa”
5: while (not end of file) do
6:   Read an integer (pattern)
7:   Search for the pattern in the pattern master file
8:   Take the 2 numerical values
9:   Find each numerical value in the master pattern file
10: end while (values are in the range of 0-255)
11: Write all the patterns found in the output file
12: Close files

```

2.4 Case Study of Compression/Decompression with Patterns

The operation we are going to perform is to compress the “Pattern” text. The first step consists of taking the ASCII code from each of the characters. The result of this step is: 80, 97, 116, 116, 101, 114, 110 and 115.

Next, pairs are taken of which only the corresponding value of the ASCII code is considered and each pair is assigned a consecutive number from 256 (it should be noted that the values of 0 al 255 correspond to the character set values of the ASCII code) as shown in Fig. 3.

P	a	t	t	e	r	n	s
80	97	116	116	101	114	110	115
256		257		258		259	

Fig. 3. Patterns generated from pairs of integer values.

The values obtained are taken, the same procedure as shown above is applied and we get what is shown in Fig. 4.

256	257	258	259	260	261
260		261		262	

Fig. 4. Other patterns generated from pairs of integer values.

Once the previous operations were performed, an integer number, which is equivalent to the corresponding couple, is obtained for each pair. The obtained numbers are stored in the pattern master file with their corresponding values as shown in Table 4.

Table 4. Patterns that are stored in the master pattern file.

Pattern	Left	Right
256	80	97
257	116	116
258	101	114
259	110	115
260	256	257
261	258	259
262	260	261

Finally, an integer equivalent to the pattern of the chain is obtained. This result is shown in Fig. 5.

Instead of the string, the number 262, which occupies only 2 bytes, is stored thus obtaining a compression rate of 96% of the original file size to be compressed i.e., the text uses 2 bytes of storage.

String	Corresponds to	Pattern
"Pattern"	→	262

Fig. 5. Text string with its respective pattern.

It should be noted that the master patterns file grows each time the compression process is performed because of this file stores all unknown combinations of byte pairs (also called patterns). Figure 6 depicts how the patterns were formed from top to bottom.

P	a	t	t	e	r	n	s
80	97	116	116	101	114	110	115
256		257		258		259	
260				261			
262							

Fig. 6. Creating patterns from top to bottom.

The second operation that is performed is decompressing the pattern 262. The decompression process, which consists of performing the opposite compression process, is applied to each pattern identified.

To clarify this process, let's take the example of compression: Be the pattern 262.

The pattern is taken and is searched in the pattern master file. This pattern is equivalent to the following values: 260 and 261. Once we have such values, the same previous process is done for each number thus obtaining the following values: 256, 257, 258 and 259.

The four numbers obtained previously are searched into the master pattern file thus obtaining the following values: 80, 97, 116, 116, 101, 114, 110 and 115.

Finally, the string obtained in the uncompressed file is written. In this way, the size of the resulting file is 2 bytes instead of 8 bytes as shown in Fig. 7.

Figure 7 depicts the complexity pyramid to decompose the patterns until recovering the original data from the file.

262							
260				261			
256		257		258		259	
80	97	116	116	101	114	110	115
P	a	t	t	e	r	n	s

Fig. 7. Pyramid of complexity of the decompression process.

3 Results and Discussion

In the investigation that was done about file compressors that exist commercially or well as shareware, can be counted by hundreds and of very different forms of operation, in various environments.

We compare the compressor operation using patterns, WinZip © and WinRar ©; in order to have comparison parameters.

They took 6 Word files that have extension. docx, they were compressed with the proposed file compressor, the WinZip © [22] and WinRar © [13]. The results and percentages obtained are shown in Table 5.

Table 5. Word files (documents) compressed with the 3 compressors.

File name	Size in bytes	Pattern compression		WinZip ©		WinRar ©	
		Size	Percentage	Size	Percentage	Size	Percentage
Pattern_Basics.docx	5,471	253	4.62%	1,696	31.00%	1,624	29.68%
Fractal.docx	9,142	397	4.34%	3,760	41.13%	3,720	40.69%
My_story.docx	12,921	549	4.25%	4,922	38.09%	4,916	38.05%
Collaboration.docx	10,567	455	4.31%	4,075	38.56%	4,033	38.17%
References.docx	9,666	419	4.33%	3,598	37.22%	4,555	47.12%
Prologo.docx	8,021	355	4.43%	3,165	39.46%	3,120	38.90%

Based on the data presented in Table 5, a comparative graph of united points was generated, obtaining the results shown in Fig. 8.

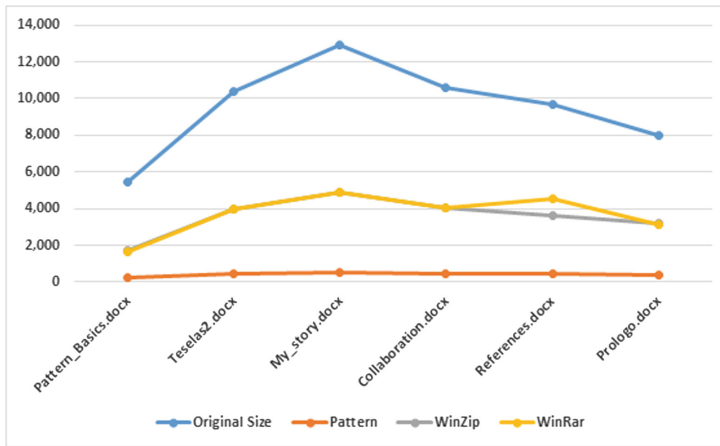


Fig. 8. Comparative graphic of compressed files with the 3 compressors.

In compressed text files, the average percentages of space they occupy are the following: Pattern compression has 4.36%, WinZip has 32.58% and WinRAR has 38.75%. Therefore the most optimal is the Pattern compression.

Another 6 image files (.BMP) were compressed with the 3 compressors that are being evaluated. The resulting percentages are shown in Table 6.

Table 6. Image files compressed with the 3 compressors.

File name	Size in bytes	Pattern compression		WinZip ©		WinRAR ©	
		Size	Percentage	Size	Percentage	Size	Percentage
Circle.bmp	7,422	335	4.5%	521	7.0%	457	6.2%
Fig410.bmp	8,702	387	4.5%	1,296	14.9%	1,161	13.3%
Arrow.bmp	7,322	331	4.5%	618	8.4%	513	7.0%
Duck.bmp	18,678	783	4.2%	1,313	7.0%	1,213	6.5%
Jlhh1.bmp	8,062	360	4.5%	819	10.2%	752	9.3%
Curve.bmp	5,862	272	4.6%	788	13.4%	708	12.1%

Based on the data presented in Table 6, a comparative graph of united points was generated, obtaining the results shown in Fig. 9.

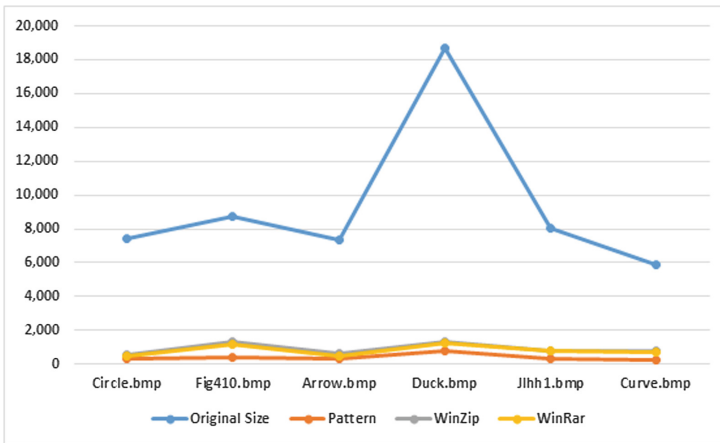


Fig. 9. Comparative graphic of compressed files with the 3 compressors (images).

In compressed image files, the average percentages of space they occupy are the following: Pattern compression occupies 4.46%, WinZip occupies 10.15% and WinRAR occupies 9.06%. Therefore the most optimal is the pattern compression.

The disadvantages of pattern compression are the following:

- A complete file folder cannot be compressed/decompressed. It is working so that in the next version, you can compress several files that are inside a folder.
- It is not compatible with WinZip and WinRAR.

4 Conclusions

In today's times, data storage devices have made dizzying progress, but also the software is becoming more complex and data files require more and more space.

In the case of executable files and any file that stores an image, sound or video, they require a large amount of space in bytes to be able to fully store the information of the same.

For many years, computer users have discussed the benefits and limitations of video cards, audio cards, memory or hard drives. However, all these components called hardware would have taken many more years to evolve without there being the need to break existing physical boundaries when handling information. It's then that it's about data compression.

As for the benefit you can use the compressor made with a certainty of 99.8%. It was used personally to compress approximately 50 files and worked correctly 100% in both their compression and decompression.

Through the development and implementation of this file compressor, new approaches, extensions, and improvements have emerged to it.

File compression and decompression may no longer be as popular today due to improvements in storage capacities, both on email accounts, hard drives, and Internet speed; in addition to the decrease of direct downloads and the growth of the torrent download (small file that contains all the information about other content that we want to download), but anyway it is still a very useful tool.

As a research project has its limitations and scopes, further work can be proposed to improve it.

Acknowledgments. Authors are grateful to TecNM/Technological Institute of Chilpancingo, Autonomous University of Guerrero (UAGro), University of Mohagheh Ardabili, University of Murcia and TecNM/CIIDET for supporting this work.

References

1. Akiyama, J.: Tile-makers and semi-tile-makers. *Am. Math. Mon.* **114**(7), 602–609 (2007)
2. Bachu, S., Chari, K.M.: A review on motion estimation in video compression. In: 2015 International Conference on Signal Processing and Communication Engineering Systems, pp. 250–256. IEEE (2015)
3. Belchor, P.M., et al.: Use of fractals channels to improve a proton exchange membrane fuel cell performance. *J. Energy Power Eng.* **9**, 727–730 (2015)
4. Bellomo, N., Bellouquid, A., Tao, Y., Winkler, M.: Toward a mathematical theory of Keller-Segel models of pattern formation in biological tissues. *Math. Models Meth. Appl. Sci.* **25**(09), 1663–1763 (2015)
5. Bentley, J., McIlroy, D.: Data compression using long common strings. In: Proceedings DCC 1999 Data Compression Conference (Cat. No. PR00096), pp. 287–295. IEEE (1999)
6. Bentley, J., McIlroy, D.: Data compression with long repeated strings. *Inf. Sci.* **135**(1–2), 1–11 (2001)

7. Kuhn, M., Kunkel, J.M., Ludwig, T.: Data compression for climate data. *Supercomput. Front. Innov.* **3**(1), 75–94 (2016)
8. Larsson, N.J.: *Structures of String Matching and Data Compression*. Lund University, Sweden (1999)
9. Lippert, L., Gross, M.H., Kurmann, C.: Compression domain volume rendering for distributed environments. *Comput. Graph. Forum* **16**, C95–C107 (1997)
10. Long, P.M., Natsev, A.I., Vitter, J.S.: Text compression via alphabet re-representation. *Neural Netw.* **12**(4–5), 755–765 (1999)
11. Makkar, A., Singh, G., Narula, R.: Improving LZW compression 1 (2012)
12. Müldner, T., Leighton, G., Diamond, J.: Using XML compression for WWW communication. In: *Proceedings of the IADIS WWW/Internet 2005 Conference* (2005)
13. RarLab, WinRar: software system for compress files (2019). <http://www.win-rar.com/rarproducts.html>. Accessed 08 Aug 2019
14. Reghbati, H.K.: Special feature an overview of data compression techniques. *Computer* **14**(4), 71–75 (1981)
15. Reghizzi, S.C., Pradella, M.: Tile rewriting grammars and picture languages. *Theor. Comput. Sci.* **340**(2), 257–272 (2005)
16. Santaolaya, S.R.: *Ambiente de Desarrollo para la Programación Visual de Interfaces de Usuario para Monitoreo de Procesos en Línea*. Ph.D. thesis, Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET) (1995)
17. Sayood, K.: *Introduction to Data Compression*. Morgan Kaufmann, Burlington (2017)
18. Sikora, T.: MPEG digital video coding standards. In: *Compressed Video over Networks*, pp. 45–88. CRC Press (2018)
19. Soria, F.G., et al.: *Sistemas evolutivos*. Boletín de Política Informática. México (1986)
20. Stamps, A.E.: Fractals, skylines, nature and beauty. *Landsc. Urban Plan.* **60**(3), 163–184 (2002)
21. SubhamastanRao, T., Soujanya, M., Hemalatha, T., Revathi, T.: Simultaneous data compression and encryption. *Int. J. Comput. Sci. Inf. Technol.* **2**(5), 2369–2374 (2011)
22. WinZip: Program of compression for windows (2019). <http://www.winzip.com/ru/prodpagewz.htm>. Accessed 08 Aug 2019
23. Wu, H., Chen, Q., Yachida, M.: Face detection from color images using a fuzzy pattern matching method. *IEEE Trans. Pattern Anal. Mach. Intell.* **21**(6), 557–563 (1999)