MultiCraft

International Journal of Engineering, Science and Technology
Vol. 1, No. 1, 2009, pp. 283-290

INTERNATIONAL JOURNAL OF ENGINEERING, SCIENCE AND TECHNOLOGY
www.ijest-ng.com

Technical note

New table look-up lossless compression method based on binary index archiving

R. Rădescu

Polytechnic University of Bucharest, 1-3, Iuliu Maniu Blvd, Sector 6, Bucharest, Romania.
E-mail: rradescu@atm.neuro.pub.ro, phone: +40-21-402-48-73; fax: +40-21-402-48-21

Abstract

This paper intends to present a common use archiver, made up following the dictionary technique and using the index archiving method as a simple and original procedure. The original contribution of the paper consists in the structure of the archived file and in the transformation of the dictionary codes into archived characters. This archiver is useful in order to accomplish the lossless compression for any file types. The application can offer important conclusions regarding the compression performances and the influence of the chosen dictionary over the parameters.

Keywords: Archive, data compression, dictionary codes, lossless algorithms

1. Introduction

The archivers, using dictionary techniques (Murgan, 1998; Rădescu, 2003a,b), can be very efficient, especially when using some files that have different words which are very often repeated. This happens because of the fact that the archivers generate their dictionaries during the archiving process, this way the program "learns" new words. Almost all the archiving programs, such as Zip, PKZip, LHArc, ARJ, GZIP, RAR etc., make use of the LZ77 and LZSS algorithms or their variants (Cover, 1991; Storer, 1998; Rao and Yip, 2001; Salomon, 2007, 2008; van Lint, 1992; Sayood, 2005; Pu, 2006; Hankerson *et al.*, 2003; Wayner, 1999, Nelson, 1991) after the files are merged together in a reversible fashion. Because the application can make an archive that contains more files, the archive has to be very well configured, so that during the unpacking of the files it can be separated with lossless information.

2. Structure of the archived file

The archived file is composed from header, followed by archived data. The header is formed from general header, followed by *n* archived file headers, where n is the number of files in the archive. For the beginning, the structure of the general header is presented (an example could be given bit by bit):

- 3 bytes to store 3 letters (CBA). These letters are used as the identification of the archive. It is very important to verify these characters in order not to let the archiver to try unpacking a file that is not a CBA archive.
- 2 bytes to store the maximum length of the dictionary.
- 2 bytes to store the minimum length of the dictionary.
- 1 byte to store the settings. This byte is used to store 3 binary validation variables:
    - 1 bit – if file has path;
    - 1 bit – if we keep the unpacked size of the file;
    - 1 bit – if file has password.

- 1 byte to store the length of the password (optional).
- *n* bytes to store the password, where *n* is the length of the password (optional).
- 2 bytes to store the number of files.

After describing the general header, the archiver repeats the following sequence for every new added archive file (archived file header):

- 2 bytes to store the length of the string that contains the path of the file (optional).
- *s* bytes for the *s* characters of the string that contains the path of the file (optional).
- 1 byte for the length of the file name.
- *f* bytes for the *f* characters of the file name.
- 4 bytes to store the unpacked size of the file (optional).
- 1 byte for the archiver type. Some files can only be copied in the archive, because their archiving would cause the increasing of the file size.
- 4 bytes for the size of the packed file.
- *nr* bytes for the *nr* characters of the packed file.

3. Packing and unpacking of the files

The process of packing and unpacking of files has 2 stages:

- the transformation of the initial characters into dictionary codes;
- the transformation of the dictionary codes into archive characters.

Transformation of the initial characters into dictionary codes

This stage is accomplished using the Lempel-Ziv-Welch (LZW) dictionary compression (Murgan, 1998; Rădescu, 2003a,b,c; Rădescu and Olteanu, 2005; Rădescu and Ene, 2005). Initially, it begins with a 257-word dictionary, i.e., the 256 ASCII characters and a special word that indicates the end of the file. The application allows a dictionary limitation. Therefore, all the values that exceed the minimum size of the dictionary will be deleted every time the maximum size of the dictionary is obtained.

The user can set both the maximum and minimum values. According to the chosen values, the number of the packed files and the compression time change. The choice of a too large maximum value of the dictionary results in a very long waiting time, getting a too small dimension improvement. The optimal values for the two limitation dictionary variables are different from one file to another.

Transformation of the dictionary codes into archive characters

This method relies on tackling from two different perspectives of two strings of numbers, having the same basic table. Dictionary codes greater than 256 elements cannot be written in the archive using only one byte. Therefore, it is necessary to have a 2 bytes space. This space is too large comparing it to the necessary one, especially in the initial phases, where the dictionary has not a large size.

From the first steps, the dictionary has a maximum of 512 elements, and the dictionary code can be written on 9 bits from the 16 available bits. Hence, 7 out of the 16 available bits remain unused, meaning almost half of the overall space. Grouping 8 codewords, 8 × 9 bits = 72 bits are needed. It can be written on 72 bits / 8 bits = 9 bytes, comparing to the 8 codewords × 2 bytes (9 bits) = 16 bytes usually needed. Even for dictionary larger than 512 elements, this method will reduce the necessary code to store dictionary codes. Table 1 refers to the transformation of codewords (the dictionary indexes) into archive characters.

Initially, it works with a 257-word dictionary (256 characters + 1 end of file control character). Table 1 is, in fact, an example in which the dictionary has a number of words ≤ 512. On the first column (423, 137, 481, 45, …) there are the codewords (dictionary indexes), which have values up to 512 that can be written on 9 bits. In order to have the certainty to obtain archive characters (8 bits), 8 codewords are used each time.

The codewords (423, 137, 481, 45, …) are binary written on the rows. This means that it will be 1byte (8bits) on each column of the table (the archive characters):

$$(8 \text{ codewords}) \times (9 \text{ bits}) = (9 \text{ archive characters}) \times (8 \text{ bits}) \tag{1}$$

The first row (161, 231, 44, 182, …) contains the character words obtained by transforming every column from binary to the 10th base. For example,

$$161 = 1\times2^7+0\times2^6+1\times2^5+0\times2^4+0\times2^3+0\times2^2+0\times2^1+1\times2^0 \tag{2}$$

It works similarly for the other values: 231, 44, 182, …

If the dictionary has between 512 and 1024 words, the procedure is similar, Table I having 8 rows, but one extra column, because every dictionary word needs 10 bits (10 columns):

$$(8 \text{ codewords}) \times (10 \text{ bits}) = (10 \text{ archive characters}) \times (8 \text{ bits}) \tag{3}$$

Table 1. Transformation of dictionary codes into archive characters

ARCHIVE CHARACTERS→ / CODEWORDS↓	161	231	44	182	14	93	152	137	241
423	1	1	0	1	0	0	1	1	1
137	0	1	0	0	0	1	0	0	1
481	1	1	1	1	0	0	0	0	1
45	0	0	0	1	0	1	1	0	1
94	0	0	1	0	1	1	1	1	0
248	0	1	1	1	1	1	0	0	0
176	0	1	0	1	1	0	0	0	0
395	1	1	0	0	0	1	0	1	1

4. Experimental results

In order to test the application, different file types are used, so that one can remark the behavior of the archive (Rădescu and Balasan, 2004; Rădescu and Popa, 2004; Rădescu and Harbatovschi, 2006; Rădescu and Balanescu, 2006; Rădescu and Bontas, 2008; Rădescu, 2009). The characteristics of the test files are presented in Table 2.

Table 2 Experimental files

File type	File no.	Min. size [KB]	Max. size [KB]	Total size [B]	Average size [B]
XLS	6	1.08	84.5	224420	37403
DOC	3	44	77.5	199680	66560
PPS	2	111	179	296960	148480
PAS	6	0.53	1.81	6247	1041
EXE	6	11.4	83.8	195050	32508
RAR	3	16.7	100	163185	54395
BMP	5	1.24	47.5	120148	24030
WAV	4	1.16	78.9	97286	24322
DLL	6	7	69	147968	24661
MID	3	21.5	39.1	86425	28808

Next, the results of the compression are shown according to the maximum size of the dictionary. For the maximum size of 512 words and the minimum size of 256 words, the compression ratio and the packing time are shown in Table 3.

Table 3 Compression ratio and compression time for the parameters (256, 512)

File type	Size [B]	Compression ratio [%]	Compression time [s]	Compression speed [KB/s]
XLS	111924	49.87	42	5.22
DOC	82044	41.09	33	5.91
PPS	257445	86.69	70	4.14
PAS	3717	59.50	2	3.05
EXE	153063	78.47	45	4.23
RAR	163185	100.00	44	3.62
BMP	70209	58.44	24	4.89
WAV	96723	99.42	26	3.65
DLL	108927	73.62	33	4.38
MID	70956	82.10	20	4.22

For the maximum size of 640 words and the minimum size of 256 words, the compression ratio and time are shown in Table 4.

Table 4 Compression ratio and compression time for the parameters (256, 640)

File type	Size [B]	Compression ratio [%]	Compression time [s]	Compression speed [KB/s]
XLS	103614	46.17	42	5.22
DOC	79726	39.93	35	5.57
PPS	264199	88.97	73	3.97
PAS	3458	55.35	2	3.05
EXE	152520	78.20	45	4.23
RAR	163185	100.00	45	3.54
BMP	68126	56.70	24	4.89
WAV	96506	99.20	27	3.52
DLL	107743	72.82	34	4.25
MID	69431	80.34	20	4.22

For the maximum size of 768 words and the minimum size of 256 words, the compression ratio and compression time are shown in Table 6.

Table 5 Compression ratio and compression time for the parameters (256, 768)

File type	Size [B]	Compression ratio [%]	Compression time [s]	Compression speed [KB/s]
XLS	97888	43.62	42	5.22
DOC	77860	38.99	36	5.42
PPS	266541	89.76	78	3.72
PAS	3414	54.65	1	6.10
EXE	151155	77.50	46	4.14
RAR	163185	100.00	48	3.32
BMP	66000	54.93	25	4.69
WAV	96292	98.98	28	3.39
DLL	106453	71.94	36	4.01
MID	67374	77.96	22	3.84

For the maximum size of 1024 words and the minimum size of 256 words, the compression ratio and compression time are shown in Table 6.

Table 6 Compression ratio and compression time for the parameters (256, 1024)

File type	Size [B]	Compression ratio [%]	Compression time [s]	Compression speed [KB/s]
XLS	90852	40.48	45	4.87
DOC	76123	38.12	39	5.00
PPS	268006	90.25	87	3.33
PAS	3396	54.36	3	2.03
EXE	150764	77.30	48	3.97
RAR	163185	100.00	53	3.01
BMP	64273	53.49	27	4.35
WAV	96056	98.74	30	3.17
DLL	15846	10.71	39	3.71
MID	66594	77.05	23	3.67

The diagrams shown in Figures 1÷4 are obtained from Tables 2÷6.

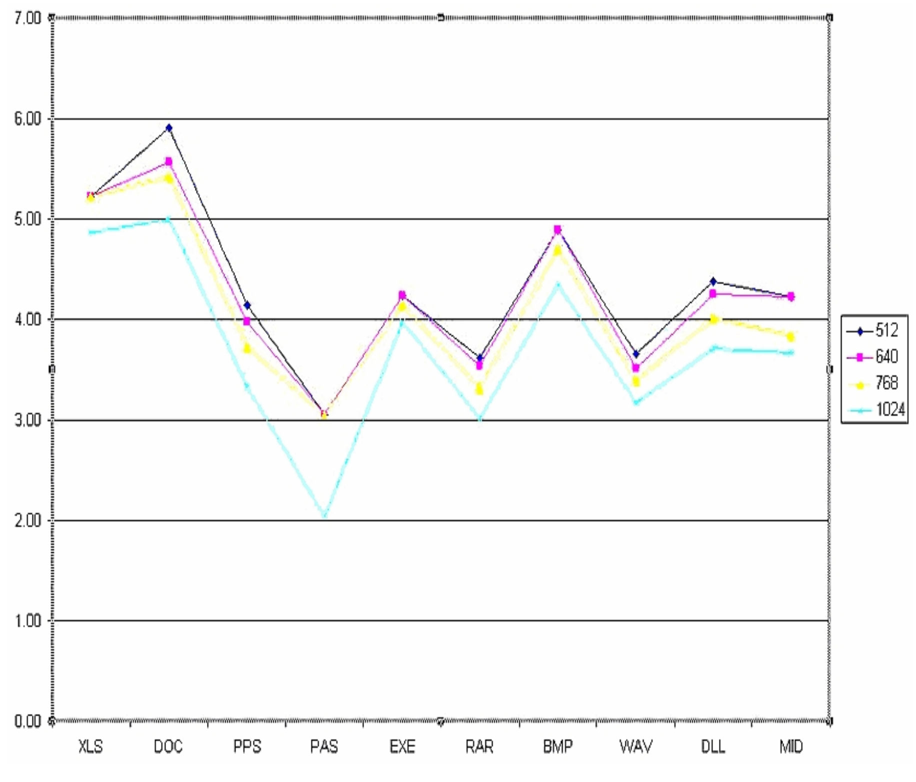

Figure 1. Compression speed [KB/s] for different file types [extensions].

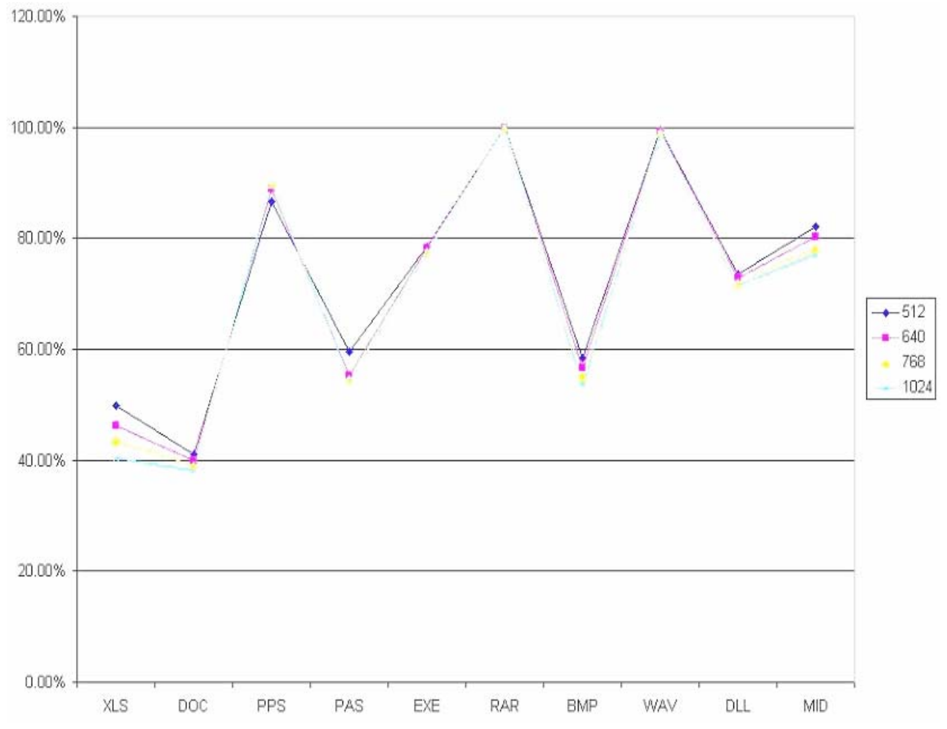

Figure 2. Compression ratio [%] for different file types [extensions].

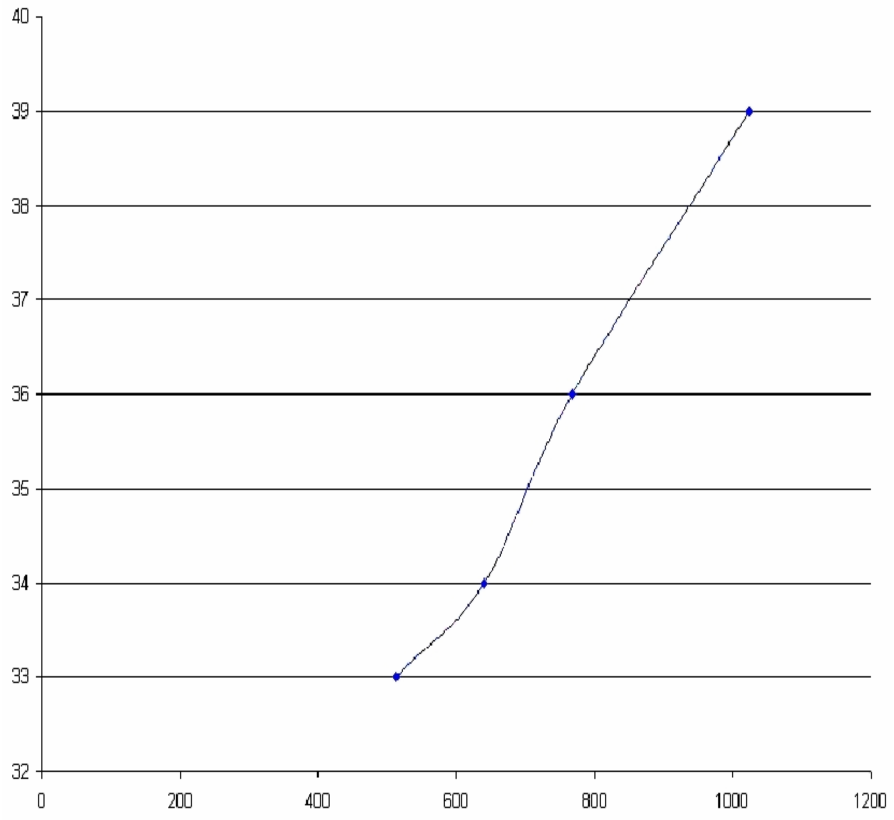

Figure 3. Compression time [s] according to the maximum size of the dictionary [KB].

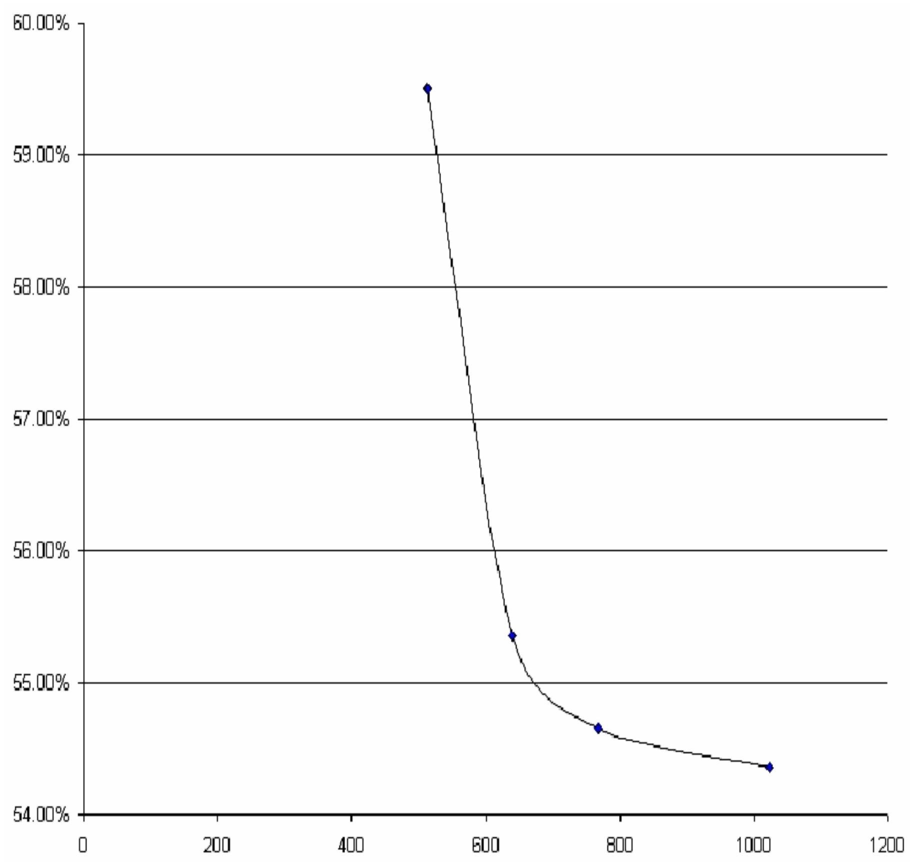

Figure 4. Compression ratio [%] according to the maximum size of the dictionary [KB].

5. Conclusion

The application described in this paper represents a good example of the way the archive performance and the waiting time are determined in the case of the alternation of the dictionary, making thus easier to understand the dictionary-based lossless compression. At the same time, the indexes archiving method can be very efficiently used not only by specialized archivers (Grupo RAR, 2009), but also in programs that manage information. This method is also recommended for storing the information for long time, where it is necessary only to check periodically the information, because of the good archiving speed.

References

Cover T.M., Thomas J.A., 1991. *Elements of information theory*, Wiley, New York.

Hankerson D., Harris G.A., Johnson P.D., Jr., 2003. *Introduction to information theory and data compression*, 2nd Ed., Chapman&Hall/CRC.

Nelson M., 1991. *The Data compression book*, IDG Books Worldwide, Inc. Foster City, CA, USA.

Murgan A.T., 1998. *Principles of information theory in information engineering and communication engineering*, Romanian Academy Press, Bucharest, (in Romanian).

Pu I. M., 2006. *Fundamental data compression*, Elsevier.

Rădescu R., 2003. *Digital information transmission – practical works*, Polytechnic Press, Bucharest, (in Romanian).

Rădescu R., 2003. *Lossless compression – methods and applications*, Matrix Rom Press, Bucharest, (in Romanian).

Rădescu R., 2003. Integrated study system of lossless data compression, *Symposium of Educational Technologies on Electronic Platforms in Engineering Higher Education*, Technical University of Civil Engineering of Bucharest, 9-10 May 2003, pp. 415-422, Conspress Bucharest, (in Romanian).

Rădescu R., Bălăşan I., 2004. "Recent results in lossless text compression using the burrows-wheeler transform (BWT), *Proceedings of IEEE International Conference on Communications* 2004 (COMM04), pp. 105-110, Bucharest, Romania, 3-5 June.

Rădescu R., Popa R., 2004. On the performances of symbol ranking text compression method, Scientific Bulletin of the "Politehnica" University of Timişoara, Romania, *Transactions on Electronics and Communications*, special issue dedicated to the Electronics and Telecommunications Symposium ETC 2004, 22-23 October 2004, Vol. 49 (63), No. 2, pp. 25-27.

Rădescu R., Şt. Olteanu, 2005. "Text and image compression using derived LZW algorithms, *EEA Revue of Electro-technique, Electronics and Automatics*, Vol. 53, No. 4, pp. 7-10, October-December (in Romanian).

Rădescu R., Ene Al., 2005. Interactive learning of lossless compression methods, *Proceedings of the Symposium "Educational Technologies on Electronic Platforms in Engineering Higher Education" (TEPE 2005)*, Technical University of Civil Engineering of Bucharest, 27-28 May, pp. 211-218, CONSPRESS Publishing House.

Rădescu R., Harbatovschi C., 2006. Compression methods using prediction by partial matching, *Proceedings of the 6th International Conference Communications 2006 (COMM2006)*, pp. 65-68, Bucharest, Romania, 8-10 June.

Rădescu R., Bălănescu C., 2006. Lossless text compression using the star (*) transform, *Proceedings of the 6th International Conference Communications 2006 (COMM2006)*, pp. 69-71, Bucharest, Romania, 8-10 June.

Rădescu R., C. Bontaş, 2008. Design and implementation of a dictionary-based archiver, *Scientific Bulletin, Electrical Engineering Series C*, University Politehnica of Bucharest, Vol. 70, Nr. 3, pp. 21-28.

Rădescu R., 2009. Transform methods used in lossless compression of text files, *Romanian Journal of Information Science and Technology (ROMJIST)*, Publishing House of the Romanian Academy, Bucharest, Vol. 12, Nr. 1, pp. 101-115.

Rao K.R., P.C. Yip (editors), 2001. *The transform and data compression handbook*, Boca Raton, CRC Press LLC.

Salomon D., 2007. *Data Compression: The complete reference*, 3rd Ed., Springer, Berlin-New York.

Salomon D., 2008. Concise *Introduction to data compression*, Springer.

Sayood K. (editor), 2005. *Introduction to data compression*. 3rd Edition, Morgan Kaufmann Series in Multimedia Information and Systems.

Storer J.A., 1998. *Data compression: Methods and theory*, Computer Science Press.

van Lint J.H., 1992. *Introduction to coding theory*, Springer, Berlin-New York.

Wayner P., 1999. *Data compression for real programmers*, Elsevier.

www.rar.com, Grupo RAR (accessed December 2009)

Biographical notes

Radu Rădescu is an Associate Professor at the Faculty of Electronics, Telecommunications and Information Technology from the Polytechnic University of Bucharest, Romania. In 1992, he registered as a member of the IEEE Information Theory Society. He became PhD in Electronics in 1998. He made traineeships at technical universities in Darmstadt, Germany (1997) and Lyon, France (1999-2000 and 2001-2002, specializing in e-learning). He prepared 15 books and manuals, 14 guides for practical works, 7 new disciplines of study, and over 110 scientific papers. He worked on 24 national and international research projects. He participated in cooperation programs with the Federal Polytechnic School in Lausanne, Switzerland, Södertörns University of Stockholm, Sweden, and Politecnico di Torino, Italy. In 2006, he was awarded the Creativity Prize at the National Conference on Virtual Learning in Bucharest. His major scientific contributions are in the fields of information theory, signal processing, e-learning systems, computer architecture, peripherals, and multimedia.

Received September 2009
Accepted October 2009
Final acceptance in revised form December 2009