Table Look-Up Lossless Compression Using Index Archiving

Radu Rădescu, *Member, IEEE*

***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. 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.**

***Index Terms*—archive, data compression, dictionary codes, lossless algorithms**

I. Introduction

THE archivers using dictionary techniques [1], [2], [3] can be very efficient, especially when using some files that have different words very often repeated. This happens because the archivers generate their dictionary during the archiving process, the program „learning" new words this way. 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.

II. Structure of the Archived File

For the beginning, the header file structure is presented:

- 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 for the path of the file;
  - 1 bit for the unpacked size of the file;
  - 1 bit for the password.
- 1 byte to store the length of the password (opt.).
- *n* bytes to store the password, where *n* is the length of the password (opt.).
- 2 bytes to store the number of files.

After describing the header of the files, the archiver repeats the following sequence for every new added archive file:

- 2 bytes to store the length of the string that contains the path of the file (opt.).
- *s* bytes for the *s* characters of the string that contains the path of the file (opt.).
- 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 (opt.).
- 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.

III. 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.

A. Transformation of the initial characters into dictionary codes

This stage is accomplished using the Lempel-Ziv-Welch (LZW) dictionary compression [1], [2], [3], [4], [7], [8]. 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. Opposed to the classical version of this method, a dictionary limitation appears in the described application. Therefore, when getting to a maximum size of the dictionary, this one will be deleted to a minimum value.

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.

R. Rădescu is with the Polytechnic University of Bucharest, 1-3, Iuliu Maniu Blvd, sector 6, Bucharest, Romania, phone: +40-21-402-48-73; fax: +40-21-402-48-21; e-mail: rradescu@atm.neuro.pub.ro.

B. *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 16 bytes usually needed. This is available for dictionary codes stored on more than 9 bits too, reducing the gained from the classical alternative.

IV. EXPERIMENTAL RESULTS

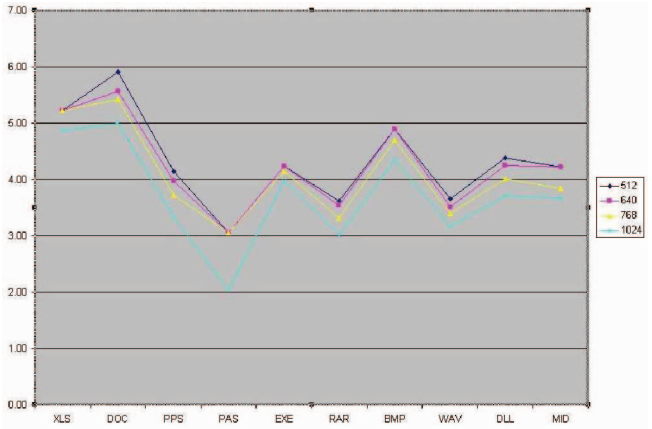

Fig. 1. Compression speed for different file types (KB/s).

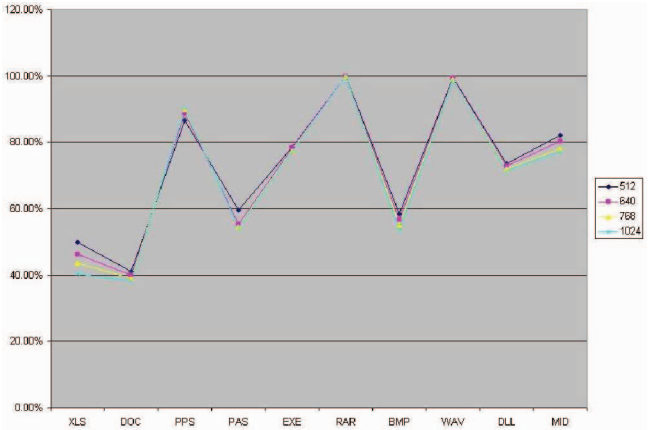

Fig. 2. Compression ratio for different file types (%).

In order to test the application, different file types are used, so that one can remark the behavior of the archive [5], [6], [9], [10].

V. 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 [11], 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

[1] A. T. Murgan, Principles Of Information Theory In Information Engineering And Communication Engineering, Romanian Academy Press, Bucharest, 1998 (in Romanian).

[2] R. Rădescu, *Digital Information Transmission – Practical Works*, Polytechnic Press, Bucharest, 2003 (in Romanian).

[3] R. Rădescu, *Lossless Compression – Methods And Applications*, Matrix Rom Press, Bucharest, 2003 (in Romanian).

[4] R. Rădescu, "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, 2003 (in Romanian).

[5] R. Rădescu, Ionuț Bălășan, "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 2004.

[6] R. Rădescu, Răzvan Popa, "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, 2004.

[7] R. Rădescu, Ştefan Olteanu, " 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 2005 (in Romanian).

[8] R. Rădescu, Alexandru Ene, "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 2005, pp. 211-218, CONSPRESS Publishing House, 2005.

[9] R. Rădescu, Cristian Harbatovschi, "Compression Methods Using Prediction By Partial Matching", Proceedings of the 6th International Conference Communications 2006 (COMM2006), pp. 65-68, Bucharest, Romania, 8-10 June 2006.

[10] R. Rădescu, Cristian Bălănescu, "Lossless Text Compression Using the Star (*) Transform", Proceedings of the 6th International Conference Communications 2006 (COMM2006), pp. 69-71, Bucharest, Romania, 8-10 June 2006.

[11] www.rar.com