International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:1, No:7, 2007

# A File Splitting Technique for Reducing the Entropy of Text Files

Abdel-Rahman M. Jaradat,, Mansour I. Irshid, and Talha T. Nassar

***Abstract*—**A novel file splitting technique for the reduction of the $n^{th}$-order entropy of text files is proposed. The technique is based on mapping the original text file into a non-ASCII binary file using a new codeword assignment method and then the resulting binary file is split into several subfiles each contains one or more bits from each codeword of the mapped binary file. The statistical properties of the subfiles are studied and it is found that they reflect the statistical properties of the original text file which is not the case when the ASCII code is used as a mapper. The $n^{th}$-order entropy of these subfiles are determined and it is found that the sum of their entropies is less than that of the original text file for the same values of extensions. These interesting statistical properties of the resulting subfiles can be used to achieve better compression ratios when conventional compression techniques are applied to these subfiles individually and on a bit-wise basis rather than on character-wise basis.

***Keywords*—**Bit-wise Compression, Entropy, File Splitting, Source Mapping.

## I. Introduction

In conventional computer-based text compression techniques, the characters in the alphabet of the input text file are transformed into an arbitrary fixed-length binary code such as the standard 8-bit ASCII code and then the compression algorithm manipulates the resulting binary text file on a character-wise basis (byte-wise basis) [1] and [2]. As it is well known, high compression ratios can be achieved by working on higher-order extensions of the original text source by finding the probabilities of the occurrence of all the combinations resulting from two or more characters. But increasing the order of extension from the $0^{th}$ order to the $n^{th}$ order requires increasing the number of combinations to $(256)^{n+1}$ which is $(256)^2 = 65536$ combinations for the $1^{st}$ order and $(256)^3 = 16777216$ combinations for $2^{nd}$ order compared to 256 combinations for the $0^{th}$ order. This mean that the higher-order extended sources requires from the compression algorithms to deal with a very large number of combinations which in turns requires very large memories and very long execution times. On the other hand, if a bit-wise source extension is used, the number of combinations can be increased in smaller jumps as we increase the order of extensions where the number of combinations is equal to 2n+1 for the $n^{th}$ order extended source, i.e. 2, 4, 8, 16, 32,… etc. It is found that a very limited research has been done on text compression based on a bit-wise basis [4].

One method based on bit-wise approach has been proposed in [3] where a non-ASCII codewords are assigned to the characters of the English text so that it would be more effective for the bit-wise run-length compression algorithm. In previous work of the authors, an efficient source encoding technique is proposed which is based on mapping the non-binary information sources with a large alphabet onto an equivalent binary source using weighted fixed-length code assignments [4], [5]. The weighted codes are chosen such that the entropy of the resulting binary source multiplied by the code length is made as close as possible to that of the original non-binary source. It is found that a large saving in complexity, execution time, and memory size is achieved when the commonly-used source encoding algorithms are applied to the bit-wise nth-order extension of the resulting binary source . This saving is due to the large reduction in the number of symbols in the alphabet of the new extended binary source where bit extensions of 1, 2, 3, 4, … can be used in the bit-wise procedure instead of bit extension of 8, 16, 24, … in the character-wise procedure.

In this paper, we propose a novel technique by which a higher compression ratios can be achieved by reducing the entropy of the text file below that of normally used values. The technique is based on mapping the original text file into a non-ASCII binary file using a new codeword assignment method which is slightly different from that used in [3] and [4]. The resulting binary file is split into several binary subfiles each contains one or more bits from each codeword of the original binary file. The statistical properties of the split files are studied and it is found that they reflect the statistical properties of the original text file which is not the case when the ASCII code is used as mapper. The $n^{th}$-order entropy of these split files are determined and it is found that the sum of their entropies is less than that of the original text file for the same extension. These interesting properties of the resulting split files can be used to achieve better compression ratios when conventional compression techniques are applied to these files individually and on a bit-wise basis rather than on character-wise basis. In section 2, the new source mapping method is discussed and the bit-wise $n^{th}$-order entropy of an English text file is found using the proposed mapping method and it is compared with that resulting from using ASCII code

Manuscript received January 26, 2006.

A. M. Jaradat is with the Electrical and Computer Engineering Department, University of Sharjah, Sharjah, P O Box 27272, United Arab Emirates (corresponding author, fax: +971-6-5050872; e-mail: jaradat@ Sharjah.ac.ae).

M. I. Irshid, is with the Electrical Engineering Department, Jordan University of Science & Technology, Irbid, 22110 Jordan.

T. T. Nassar is with the Computer Science Department, Jordan University of Science & Technology, Irbid, 22110 Jordan.

mapping.

In section 3, the proposed splitting technique is discussed and the bit-wise $n^{th}$-order entropy of an English text file is found for the individual split files.

Conclusions are given in section 3.

## II. Source Mapping

In conventional text compression techniques, the original text file is transformed into a binary file using the standard ASCII code in which the 8-bit binary codewords were assigned to the different text characters without given any attention to the statistical properties of the individual bits of the codewords. Such assignment method allows the compression technique to work on the resulting binary file on a character-wise basis only and in this case the compression ratio will be independent of the way the codewords are assigned to the original characters. On the other hand, if a bit-wise compression technique is to be used then the mapping of the original text file into binary file must be done in a specific procedure such that the statistics of the original characters of the source are reflected onto the statistics of the two binary symbols in the resulting binary file. The optimum mapping method is the one which makes the entropy of the resulting $0^{th}$ -order binary file as close as possible to the entropy per bit of the original file.

In previous work, the authors successfully applied a new source mapping technique to compress text files using conventional compression techniques but working on the binary file on a bit-wise basis rather than on a character-wise basis [4], [5] and recently in [6] it was extended to the compression of multimedia files.

In this paper, the proposed mapping technique is based on mapping each character in the original text file into an 8-bit codeword based on the frequency of occurrence of the original character. The mapping is done by arranging the different characters of the alphabet of the given text file in a descending order of their probabilities of occurrence, and then the codewords are given to the different characters such that each codeword is the 8-bits binary representation of the position of that corresponding character in the list.

The most frequent character is represented by an 8-bit binary sequence of all zeros (which is the binary representation of decimal zero), the second most frequent character is represented by all zeros but a one in the LSB of the binary sequence (which is the binary representation of decimal one) and so on. A source mapper for the English language (256 symbols) is designed according to the proposed mapping rule and the probabilities and the assigned codewords for the different characters are shown in Table I (a sample of the characters is shown in the table).

TABLE I
PROBABILITIES AND THE NEW CODE ASSIGNMENTS OF THE ENGLISH TEXT ALPHABET

| Character | Probability | Assigned code |
|---|---|---|
| Space | 0.174 | 00000000 |
| e | 0.098 | 00000001 |
| t | 0.070 | 00000010 |
| a | 0.062 | 00000011 |
| o | 0.059 | 00000100 |
| i | 0.055 | 00000101 |
| n | 0.055 | 00000110 |
| s | 0.050 | 00000111 |
| r | 0.048 | 00001000 |
| h | 0.042 | 00001001 |
| ... | ... | ... |

Let the original information source ***S*** has an alphabet of $M = 2^N$ symbols with probabilities $p_0$, $p_1$, $p_2$, …, $p_{M-1}$, and let the equivalent binary source ***B*** has probability $p_0$ for symbol 0 and probability $p_1 = 1 - p_0$ for symbol 1. The $0^{th}$-order entropy of the original text file is calculated on a character-wise basis using the following well-known entropy equation:

$$H_C = -\sum_{i=0}^{255} p_i \log_2 p_i \quad (1)$$

When substituting the probabilities of the different characters of the English language given in Table I in (1), it is found that the entropy of the original text source is 4.47 bits/character [7]. The $0^{th}$ entropy of the binary file which results from mapping the original source using the above proposed mapping method can be found by calculating the probabilities of symbol 0 and symbol 1 of the resulting binary source are found to be

$$p_0 = 0.8, p_1 = 0.2 \quad (2)$$

The entropy of the resulting $0^{th}$-order binary source is found by substituting the probabilities of 0 and 1 in the following binary entropy equation:

$$H(B) = -p_0 \log_2 p_0 - p_1 \log_2 p_1 \quad (3)$$

and it is found to be 0.7219 bits/symbol. The entropy for n bits of the binary file is n times that of its $0^{th}$-order entropy, i.e., 0.7219 n bits/symbol. The entropy of the 8 bits of the binary source which is equivalent to one character of the original file is found to be 5.7754 bits/character compared to 4.47 bits/character calculated on character-basis for the original text file.

For comparison purposes, the entropy of the binary source resulting from using the standard ASCII code mapper is calculated and it is found that the probabilities of symbol 1

and symbol 0 are 0.44 and 0.56, respectively and the entropy of the $0^{th}$-order binary source is 0.99 bits/symbol and the entropy of the 8 bits which is equivalent to one character is 7.9 bits/symbol which is much greater than the entropy of the original source. This comparison indicates that bit-wise compression technique cannot achieve any compression when applied to binary sources using ASCII code or any other random code. The $n^{th}$-order entropy for the binary source resulting from the proposed mapping method and that resulting from the ASCII code is calculated for various values of bit extensions where n takes values from 1 to 32 where a computer program is written to determine these entropies by finding the probabilities of occurrence of the $2^n$ bit combinations in the binary file. Fig. 1 shows the nth entropy as function of bit extensions for an English text file using the proposed source mapper and the ASCII code mapper.

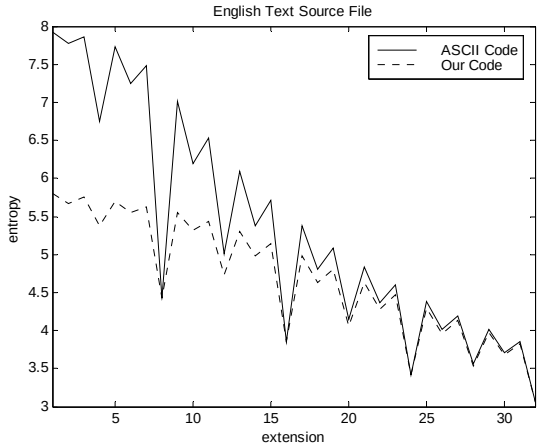


Fig. 1 The $n^{th}$-order entropy of the English text file for the proposed Code and the ASCII Code for various bit extensions

As it is shown from this figure, the two mappers give the same $n^{th}$ entropy for the 8, 16, 24, and 32 bit extensions and this result is as expected since these extensions correspond to the $0^{th}$, $1^{st}$, $2^{nd}$, and $3^{rd}$ extensions based on character-wise entropy calculations which are independent of codeword assignments. For bit extensions which are not multiples of number 8, the entropy of the ASCII code is quite larger than that of the proposed code specially for lower extension values but the difference decreases as the bit extension increases. For bit extensions lower than eight, there is a saving of approximately two bits in the value of the entropy when using the proposed mapping method. This saving in entropy at low bit extensions can help in designing simple compression techniques with large saving in complexity, execution time and memory requirements [4], [5].

## III. File Splitting

In [4], [5], the compression techniques are applied to the whole binary file which results from mapping the original text file using the proposed mapping method. Although, the compression achieved in these papers is comparable to that of conventional methods but they have the advantage of large saving in complexity, execution time and memory size which is due to the large reduction in the number of symbols used in the compression process. In this paper, a novel method is proposed to achieve more compression by reducing the $n^{th}$ order entropy of the text file. This reduction of the entropy is achieved by splitting the mapped binary file into several subfiles and the total entropy of the file is found by determining the entropy of the subfiles and summing them up. To achieve compression ratios below that of conventional methods, the compression algorithm has to be done on the subfiles rather than on the whole binary file.

By examining the variation in the probabilities of the binary symbols in the different bit position, we find that the probabilities of symbol 1 and symbol 0 in the least significant bit of the codewords are almost equal while the difference between these two probabilities start to increase as we move toward the most significant bit where it is almost one for symbol 0 and almost zero for symbol 1. This means that the entropy of the LSB is nearly equal one while it is nearly equal zero for the MSB and has values between zero and one for the rest of the bit positions. This interesting fact leads us to device a technique which uses this wide variations in the entropy of the different bit positions in the binary text file. The technique is based on splitting the binary file into two, four or eight subfiles each of them contains equal parts of the original file based on dividing the bits of the codewords in specified manner.

For example, in the case of splitting the binary file into two subfiles, the four most significant bits (7, 6, 5, and 4) of each codeword in the binary file are stored in one subfile while the four least significant bits (3, 2, 1 and 0) are stored in the other subfile where each subfile has a size half that of the original binary file. While in splitting the original binary file into 4 subfiles, the following bit pairs (7, 6), (5, 4), (3, 2), and (1, 0) of each codeword are stored in the four different subfiles where each subfile has a size one fourth the size of the original binary file. While in the case of eight subfiles, the bits of each bit position in the codeword is stored in a different subfile, i.e. the MSB is stored in one subfile, the next significant bit is stored in another subfile and the same thing is done for the rest of bits resulting in eight subfiles each has a size one eighth the size of the original binary file. The $0^{th}$-order entropy of each subfile can be calculated mathematically by calculating the probabilities of symbol 0 and symbol 1 from the probabilities of the original characters of the text file. The probability of symbol 0 in the $7^{th}$ bit position of the proposed mapped code is equal to the sum of the probabilities of the characters whose MSB is zero and it can be determined by the following equation:

$$p_{07} = \sum_{i=0}^{127} p_i \tag{4}$$

Similarly, the probability of symbol 0 in the $6^{th}$ and the $5^{th}$ bit positions are given by the following equations:

$$p_{06} = \sum_{i=0}^{63} p_i + \sum_{i=128}^{191} p_i \quad (5)$$

$$p_{05} = \sum_{i=0}^{31} p_i + \sum_{i=64}^{95} p_i + \sum_{i=128}^{159} p_i + \sum_{i=192}^{223} p_i \quad (6)$$

The equations for determining the probability for symbol 0 for the rest of the bit positions can be found in a similar procedure but because of the large number of summations they are not shown here. The 0th order entropy of the nth bit position can be determined by using the following binary entropy equation:

$$H_n = -p_{0_n} \log_2 p_{0_n} - p_{1_n} \log_2 p_{1_n} \quad (7)$$

where $p_0$ is the probability of symbol 0 and $p_1 = 1 - p_0$ is the probability of symbol 1.

The overall entropy of the mapped binary file can be found by adding the entropies of the eight bit positions:

$$H_T = \sum_{i=0}^{7} H_i \quad (8)$$

The probabilities of symbol 0 and symbol 1 and the corresponding entropies for the various bit positions are shown in Table II.

TABLE II
THE PROBABILITIES AND THE ENTROPIES OF THE DIFFERENT BIT POSITIONS OF THE PROPOSED CODEWORDS IN AN ENGLISH TEXT FILE

| Bit # | $P_0$ | $P_1$ | Entropy |
|---|---|---|---|
| 7 | 1.000 | 0.0000 | 0.0000 |
| 6 | 0.999 | 0.0010 | 0.0114 |
| 5 | 0.966 | 0.0340 | 0.2141 |
| 4 | 0.883 | 0.1170 | 0.5207 |
| 3 | 0.731 | 0.2690 | 0.8400 |
| 2 | 0.636 | 0.3640 | 0.9460 |
| 1 | 0.590 | 0.4100 | 0.9765 |
| 0 | 0.553 | 0.4470 | 0.9919 |

The total 0th-order entropy $H_T$ of the mapped binary file when it is split into 8 subfiles is found to be 4.818 bits/character compared to 5.775 bits/character for the non-split binary file. This entropy value is almost equal to the 0th-order entropy of the original text file calculated on character-wise basis which is 4.47 bits/character. This is a very encouraging result since we can device simple compression techniques working on a bit-wise basis and achieving compression ratios comparable to that working on character-wise basis but with much lower symbol combinations. It is obvious from Table II that the four least significant bits have the largest contribution to the overall entropy of the mapped binary file while the four most significant bits have the lowest contribution where the contribution from the four LSB is 3.75 bits/character and from the four MSB is 0.75 bits/character. Since it is very difficult to determine the nth order entropy of the subfiles mathematically, a computer program was written to determine these entropies by finding the probabilities of the $2^n$ bit combinations in the each subfile and then substituting the results in the entropy equation. The effective entropy of the whole split binary file is equal to the sum of the individual entropies of the subfiles.

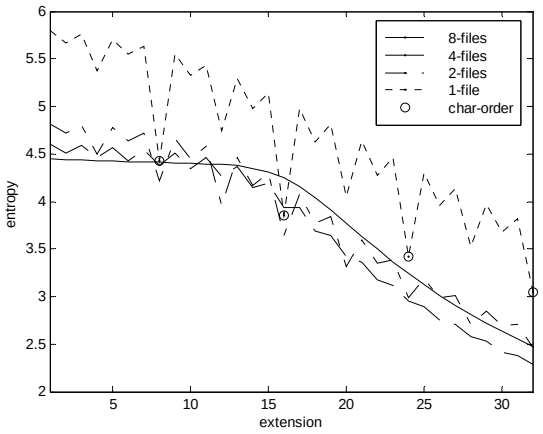


Fig. 2 The nth-order entropy of the split English text file as function of bit extensions using the proposed Code for the various splitting cases

Fig. 2 shows the overall entropy of the mapped binary file as function of bit extensions for one, two, four and eight subfiles where the case of one subfiles is that of the non-split mapped binary file. It is obvious from this figure that the three cases of split files give overall entropy which is much less than that of the non-split case except for bit extensions which are multiple of number eight which is the length of character codeword. Also it is obvious that as the number of subfiles increases the entropy curve becomes more smother. For bit extensions less than eight, the 8 subfile case gives the lowest entropy which is nearly constant and it is almost equal to the 0th-order character-wise entropy of the original text file and this interesting property can be utilized to device a simple method for determining the 0th order character-wise entropy for sources having large number of alphabets. The two-subfile case has the best entropy properties for values of bit extension lying between 8 and 16 where its entropy at the 8 and 16 bit extensions is less than that of the corresponding 0th and the 1st character-wise entropy of the original text file. Moreover, the entropy of the two-subfile case at the 12 bit extension is equal to 4 bit/character compared to 4.75 bit/character for the non-split binary file and compared to 4.47 bit/character for the 0th order character-wise entropy and compared to 3.86 bit/character for the 1st order character-wise entropy. This means that a good compression can be achieved at the 12 bit extension of the two-subfile case where the compression

algorithm deals with $2^{12}$=4096 binary combinations compare with $2^{16}$=65536 binary combinations required for the 16-bit extension (1st-order character-wise extension ).

The four-subfile case has the best entropy properties for values of bit extension greater than 16 where it has the least values of entropy compared with the other cases. This case has an entropy of 3 bit/character at the 24 bit extension compared with 3.42 bit/character for the corresponding 2nd-order character-wise extension and an entropy of 2.3 bit/character at the 32 bit extension compared with 3.05 bit/character for the corresponding 3rd-order character-wise extension. The reduction in the entropy of split file below that of the original text file is due to the fact that when extending the different subfiles by n-bits, the actual extension in the original text file is 2n, 4n, and 8n bits for the two, four, and eight subfiles cases.

For comparison purposes, Fig. 3 shows the nth-order entropy of the split English text file as function of bit extensions using the proposed Code and the ASCII code for the 8 subfile splitting cases which shows that this file splitting technique works only with specific codeword assignment methods similar to that proposed in this paper.

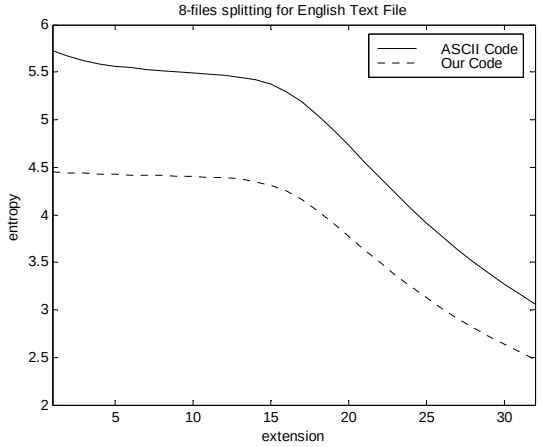


Fig. 3 The nth-order entropy of the split English text file as function of bit extensions using the proposed Code and the ASCII code for the 8 subfile splitting cases

## IV. Conclusion

In this paper, a new technique is proposed by which the nth -order entropy of text files can be reduced below that of its values calculated on a character-wise bases. The is achieved via mapping the original text file into a non-ASCII binary file using a new codeword assignment method and then the resulting binary file is split into several binary subfiles each contains one or more bits from each codeword of the mapped binary file. The nth-order entropy of these subfiles are determined and it is found that their sum is less than that of the original text file for the same values of extensions. The reduction in the entropy of the resulting subfiles can be used to device compression algorithms with better compression ratios. This can be done by applying the conventional compression techniques to the subfiles individually and on a bit-wise basis rather than on character-wise basis. By applying the technique to the same text file but with ASCII mapping, it was found that this file splitting technique works only with specific codeword assignment methods similar to that proposed in this paper.

## References

[1] T. C. Bell, J. G. Cleary, and I. H. Witten, *Text Compression*, Prentice-Hall, Englewood cliffs NJ, 1990.
[2] G. Held and T. R. Marshall, *Data Compression*, John Wiley, New York, 1991.
[3] M. F. Lynch, "Compression of bibliographic files using an adaptation of run-length coding," Information Storage and Retrieval, Vol. 9, pp. 207—214, 1973.
[4] A. M. Elabdalla and M. I. Irshid, "An efficient bitwise Huffman coding technique based on source mapping," Computers and Electrical Engineering, Vol. 27, pp. 265—272, 2001.
[5] A. M. Jaradat and M. I. Irshid, "A simple binary run-length compression technique for non-binary sources based on source mapping," Active and Passive Electronic Components, Vol. 24, pp. 211—221, 2001.
[6] A. A. Sharieh, "Enhancement of Huffman coding for the compression of multimedia files," International Journal of Information Technology, Vol. 1, pp. 211—213, 2004.
[7] Calgary-corpus: ftp://ftp.cpsc.ucalgary.ca/pub/projects/text.compression.corpus