A Bandwidth Efficiency of the Locally Adaptive Compression Method for Hiding Secret Messages

Tzu-Chuen Lu [1] Chin-Chen Chang [2]

[1] Department of Information Management,
Chaoyang University of Technology, Taichung, Taiwan, 41349, R.O.C.
E-mail:tclu@cyut.edu.tw
[2]Department of Information Engineering and Computer Science,
Feng Chia University, Taichung, Taiwan, 40724, R.O.C.
E-mail: ccc@cs.ccu.edu.tw

Abstract

This paper proposes a novel lossless information-hiding scheme that hides information in locally adaptive compressed codes. The scheme is abbreviated as IHLAC for wider the efficiency of the compressed codes. IHLAC is designed for two objectives; one is to compress a text file; the other is to hide secret information in it. According to the experimental results, IHLAC can easily and efficiently reach the objective of hiding huge secret information. IHLAC can exactly decompress the original text file and extract the secret information. Only a small number of extra bits are needed to record the hidden information. In addition, the IHLAC can be even used to hide secret information in image files without distorting the image quality.

Keywords: data compression, information hiding, locally adaptive.

1. Introduction

Locally adaptive scheme (LAS) is a popular lossless data compression method that provides satisfactory compression rate for text file. In terms of the text-compressed process of LAS, we notice some useless compressed codes in the text file. If those codes are wisely used to hide information, then we not only send the text file to a receiver in the short time but also share secret information to the receiver. However, hiding information in the compressed codes may increase the bandwidth efficiency of the compressed codes. The initial idea of hiding information within a compressed representation of text was proposed by Cachin who modifies Willems' algorithm to combine information hiding and data compression [6]. Then Atallah and Lonardi proposed an LZS-77 algorithm to hide secret message within the Lemple-Ziv compressed representation of a document [2]. The length of the secret message that LZS-77 algorithm can embed in a text file grows linearly with the length of the text file. The LAS compressed codes can be used to hide information as well as the Lemple-Ziv can hide information. Therefore, this paper will propose a new algorithm to hide information within the LAS compressed codes.

In order to hide the information in the compressed codes more efficiently, we need advanced information hiding method. Many researchers have proposed their methods for hiding information in variety of host signals, such as texts, images, sounds or audios [3, 7]. The compressed codes can be regarded as texts to hide information, by means of the compression methods to turn the original file into it corresponding binary string. In this way, we may apply the information hiding methods to embed information in the compressed codes. Some researchers regarded text files as image files that can be used to hide information [5]. Although most of the mentioned methods will result in indiscernible changes to the human eye, they still have to modify the contents of the text file.

This paper shall propose a novel concept that hides information and compresses a text file at the same time. In addition, the proposed method will not modify the contents of the secret information and the text file.

2. Data Compression and Information Hiding

The objective of a data compression algorithm is to achieve the smallest compressed file within the same text. Text compression is not suitable for image compression, which is applied to lossy compression. That's why text files are usually compressed using lossless techniques. The original data can be exactly recovered from the compressed codes by lossless compression technique. There have been many

classical lossless text compression algorithms proposed, such as Huffman encoding, Lempel-Ziv encoding, run length encoding, arithmetic coding, move to front encoding, and so on.

Huffman encoding is an optimal coding method, which is composed of three steps, a count of character frequencies, the construction of the prefix code, and the encoding of the text. Although Huffman encoding gives the optimal code for compressing a file, it also has two fatal defects. One is that it needs to scan the source text file twice if the frequencies of characters are unknown. The other is that the coding tree must be included in the compressed file [1].

In order to solve the problems of Huffman encoding, an Adaptive Huffman Coding scheme was proposed. It only requires one pass to compress data, which determines the mapping from source characters to the compressed codes based upon a running estimate of the source characters probabilities. The other one-pass lossless data compression algorithm is the move-to-front data compression algorithm (MTF), which assigns codes with lower values for more redundant characters. The preprocessing step of MTF is to translate all the characters into a move-to-front list. When a character is processed, MTF outputs a code, which refers to the position of the character in the list, instead of outputting the character, and then puts the character at the top of the list. MTF provides a simple method to compress text. Nevertheless, the compression rate of MTF is relatively poor.

LAS encoding technique is proposed to improve the compression rate of MTF. LAS also haves a list to record the occurring characters. The most obvious difference between LAS and MTF is that the list in LAS is an empty set in the initial phase. In addition, LAS needs an indicator to indicate whether the characters exist in the list, where 0 means non-existence and 1 means existence. While the character in the input file does not exist in the list, the character will be put at the top of the list, and then a binary string of the character is output [8].

For example, assume the input string is "bbbbaaaa," the compressed string is "$\underline{0}(b)_2$ $\underline{1}0$ $\underline{1}0$ $\underline{1}0$ $\underline{0}(a)_2$ $\underline{1}0$ $\underline{1}0$ $\underline{1}0$," where $(b)_2$ is the binary string of b, and the underlined compressed codes are indicators. Because there are only two distinct characters 'a' and 'b' in the string, we only use one bit to represent the compressed code of each character. If the number of distinct characters in the string is 3, then the length of the compressed codes is 2. The useable compressed codes are 00, 01, 10 and 11. However, the compressed code 11 is useless in this case.

The useless codes can be used to represent the hiding information for increasing the bandwidth efficiency of the compressed codes. In order to embed information in the useless compressed codes, an efficient information hiding method is also required. Many researchers have proposed their methods for hiding information in text files. Some of them detected features of a text for encoding a document, such as the vertical shift of lines in a text or the horizontal shift of words within text lines, or chosen and altered text features [4, 9, 11].

These methods have some inherent problems, such as inefficiency, the requirement for a great deal of text to encode a very few bits, or the impossibility of retrieving the hidden data from a hard copy. The ability to encode the text file is dependent on the structure of the text, such as the amount of spaces or tabs in the text file. Most of the methods need a source text to extract the hidden information, or they will modify the contents of the text file. Nonetheless, a text file is a discrete information source such that the contents cannot be modified.

In this paper, we shall propose an efficient information hiding scheme that is based on the LAS compression technique. The proposed scheme quickly compresses a huge amount of data into small-size compressed codes and effectively hides important information in the compressed codes that without modifying the contents of the original file. Like other adaptive encoding techniques, IHLAC does not require a first pass to analyze the characteristics of the source. In addition, IHLAC provides encoding and transmission in real time.

3. The Proposed Method

In this section, we are going to propose a novel information-hiding scheme, which is called IHLAC. IHLAC directly hides information in the compressed codes and extracts the hidden information from the compressed codes.

Assume an input string is composed of a sequence of characters. The goal of data compression is to compress the long input string into smaller compressed codes. Let S be the input string and H be a secret information string which is prepared to embed in the input string. H is obtained by encrypting a plaintext using a DES-like method associated with the private key. Because certain characters often occur together or in proximity, we make a list, called a basin, to record the previously occurring characters. Let B be the basin that records the history characters of S, where the number of elements of B is K.

In LAS, the length of compressed codes of a character which existed in the basin is computed by $\lceil \log_2 K \rceil$. This means that if the number of alphabet characters in B is less than 2, $K \leq 2$, then only one bit is needed to represent the first and second characters

in *B*. If the size of *B* is not a multiple of a number that is a power of 2, then some unused codes will remain. For example, if *K*=5, we use three bits to represent each character in *B,* since $3 = \lceil \log_2 5 \rceil$. The usable codes to represent a character are 000, 001, 010, 011, 100, 101, 110 and 111. However, the codes 101, 110 and 111 are unused. In this paper, we shall use the unused codes to hide information.

Nonetheless, if *K* is a multiple of a number that is a power of 2, then no unused code can be used. Therefore, this paper defines a new encoding strategy to represent the character, and it allows information to be hidden in any size of *K*.

Let x be the index of an input character in the *B*.

(1) If $\lceil \log_2(K+1) \rceil > \lceil \log_2(x+1) \rceil$, then converts x to binary format and pad the string with zeroes so that it has the appropriate length.

(2) If $\lceil \log_2(K+1) \rceil \leq \lceil \log_2(x+1) \rceil$, then converts x to binary format.

For example, if *K* is a multiple of a number that is a power of 2, then we increases one bit to represent the compressed codes. For example, assume *K* = 2, which is a multiple of a number that is a power of 2. We use two bits to represent the compressed code of a character. In this case the unused codes 10 and 11 can be used to hide information. Although the total length of compressed codes from IHLAC is greater than that from LAS, IHLAC can hide more information in the compressed codes than LAS can. The unused codes in different sizes of *K* are show in Table 1.

Table 1 The encoding diagram in IHLAC

K	size	Used Codes	Unused Codes
1	1	0	1
2	2	00, 01	10, 11
3	2	00, 01, 10	11
4	3	000, 001, 010, 011	100, 101, 110, 111
5	3	000, 001, 010, 011, 100	101, 110, 111
…	…	…	…

There are three cases to embed information in the compressed codes.

Case 1: The character does not exist in the basin. Output the binary string of the character.

Case 2: If the character exists in the basin and the index of the character has the same parity as the information, then we can embed information in the compressed code. In this case, output the compressed codes of the character directly.

Case 3: If the index of the character does not the same as the information, then output the unused codes to represent the information. Furthermore, check next information to see if it has the same parity as the index. Repeat Case 3 until the information has the same parity as the index.

For example, assume there are three characters existed in *B*, where *B* = <*a, b, c*>. An input character 'a' is existed in the first location. The index of the character is 0. Suppose that the information is 0. The index of the character has the same parity as the information. Hence, we output the compressed code of the character directly. On the other hand, assume that the input character is 'c' and the information is 1. The index of 'c' is 2. The index has difference parity with the information. Thus, the encoder outputs the unused codes 11 to represent the information, and checks next information to see if it has the same parity as the index.

While the receiver receives the compressed results, the decoder is used to decompress and extract the original characters and the hidden information.

If the indicator is 1, then the decoder scans next seven characters to form a binary string and coverts the string into an ASCII character. The converted character is then added into the basin *B*.

In the other hand, if the indicator is 0, then the decoder scans next $\lceil \log_2 K \rceil$ characters to form a binary string, where *K* is the number of characters in *B*. Then, the decoder converts the string into an integer, which is the index of the original character.

If the converted index is greater than the size of *B* that means the compressed code of the original character has different parity with the information. The converted index only represents the information but the original character. As for the value of the information, it is depended on the value of next index. The decoder continually checks next index until the index is less than the size of *B*. If the final index is odd, then the previous information is even. Otherwise, the pervious information is odd.

4. Experiments

The proposed scheme was tested on uncompressed text and image files. Several text files in the standards were used for compressing and hiding to evaluate the performance of the proposed scheme. The standards are the Canterbury Corpus, the Artificial Corpus, the Large Corpus, the Miscellaneous Corpus, and the Calgary Corpus [10].

The experiment was to compare the performance of IHLAC with other hiding methods SNOW, and Wbstego4. A steganographic nature of whitespace (SNOW) program was proposed by Kwan to conceal messages in an ASCII text by appending white spaces at the end of lines, and to extract messages from files containing hidden messages. Because spaces and tabs are generally not visible in text viewers, the message is effectively hidden from casual observers [9]. WbStego

4 hides data in bitmap images, ASCII and ANSI text files, HTML files, and PDF files. The hidden file is not optically changed. It can be used to exchange sensitive data securely or to add hidden copyright information to the file [11].

Table 2 shows the compression results of the IHLAC, SNOW, and Wbstego 4. Some results of SNOW shown in Table 2 are labeled with "-." This is because the size of the test files was too long to handle, or the spaces of the files were not enough to hide information, so the decompressed text files were incomplete.

The sizes of the output results by using Wbstego were the same as that of the original text file. Nevertheless, the information load of the Wbstego is dependent on the spaces of the test files. The maximum size of information accepted by the Wbstego approximately equals 1.25% of spaces in the test file.

According to the experimental results, the performance of the proposed scheme outperforms other schemes.

Table 2 The compression results between IHLAC and other methods

File name	SNOW	Wbstego 4	IHLAC
bib	105,033	104,982	104,905
book1	-	752,150	751,857
book2	595,271	595,223	594,849
news	-	367,049	366,965
paper1	51,973	51,912	51,837
paper2	80,530	80,469	80,403
paper3	45,488	45,427	45,292
paper4	13,021	12,993	12,802
progc	38,170	38,125	38,039
progl	69,449	69,403	67,985
trans	-	88,315	88,793
aaa	100,062	100,001	12,502
random	100,058	100,001	87,518
alphabet	100,062	100,001	75,013
alice29	144,938	144,874	143,779
Asyoulik	121,117	121,058	120,083
lcet10	411,496	411,717	411,064
plrabn12	-	460,464	460,135
Bible	-	4,017,010	3,514,854
E.coli	4,638,756	4,638,691	1,739,515
pi	1,000,066	1,000,001	625,006

5. Conclusions

This paper proposes a novel lossless information-hiding scheme that is based on a locally adaptive compression technique, called IHLAC. IHLAC provides higher hiding capacities and allows complete recovery of the original host signal and hidden information. The average time of the IHLAC system needed to compress a character is 5.1337 microseconds. From the experimental results, it is obvious to see that IHLAC is indeed an efficient and effective scheme for compressing host signal and hiding secret information.

6. References

[1] Atallah, J. (1998) *Handbook on Algorithms and Theory of Computation*. CRC Press, Florida.

[2] Atallah, M. J., Lonardi, S. (2003) Authentication of LZ-77 Compressed Data. Proceedings of the ACM Symposium on Applied Computing, pp. 282-287.

[3] Bender, W., Gruhl, D., Morimoto, N. and Lu, A. (1996) Techniques for Data Hiding. *IBM Systems Journal*, Vol. 35, No. 3&4, pp. 331-336.

[4] Brassil, J.T., Low, S.H., Maxemchuk, N.F. and O'Gorman, L. (1995) Electrical Marking and Identification Techniques to Discourage Document Copying. *IEEE Journal on Selected Areas in Communications*, Vol.13, No. 8, pp. 1495-1504.

[5] Brassil, J.T., Low, S.H., Maxemchuk, N.F. and O'Gorman, L. (1995) Hiding Information in Document Images. *Proceedings of the 29th Annual Conference on Information Sciences and Systems*, Baltimore, Maryland, pp 482-489.

[6] Cachin C. (1998) An Information-Theoretic Model for Steganography. Proceedings of the 2nd Workshop on Information Hiding, Lecture Notes in Computer Science, Springer, 1998.

[7] Celik, U., Sharma, G., Tekalp, A.M. and Saber, E. (2002) Reversible Data Hiding. *Proceedings of IEEE International Conference on Image Processing*, Rochester, New York.

[8] Chang, C.C. and Wang, C.H. (1997) A Locally Adaptive Data Compression Strategy for Chinese-English Characters. *Journal of Systems and Software*, Vol. 36, Issue 2, pp. 167-179.

[9] The SNOW software can be found from the SNOW homepage or explicitly at http://www.darkside.com.au/snow/.

[10] The test files courtesy of T. Bell and M. Powell may be found at http://corpus.canterbury.ac.nz/index.html.

[11] The WbStego 4 software can be found from the Evidence Eliminators homepage or explicitly at http://evidence-eliminators.co.uk/wbstego_4.htm.