Journal of University of Babylon, Pure and Applied Sciences, Vol.(25), No.(5), 2017.

Adapted LZW Protocol for ECG Data Compression

Saif M. Kh. Al-alak saif.shareefy@gmail.com
I. H. Alwan isr.phd@gmail.com
Ahmed A. Hussein aahussein38@gmail.com
Babylon University, College of science for women, Dept. of Computer science

Abstract

Lempel–Ziv–Welch (LZW) is a data compression method, which is adopted by many applications likes Electrocardiography (ECG) data to reduce the size of transferred data. Because of the ECG data moves over the network all the time, which means there is a need to reduce its size to improve the network performance. In this paper, we concerned with the LZW method, which is one of the important and famous data compression method. We propose a protocol to improve the way in which the LZW saving an index for the compressed data. The proposed protocol could reduce the size of the index in LZW method. Five samples data groups provided by Physionet are used for evaluation. The experimental result shows that the proposed protocol can give best compression ratio compared with the original method.

Keyword: LZW,ECG ,Data Compression, Useless compression

الخلاصة

خوارزمية الـ(LZW) هي واحدة من طرق ضغط البيانات المستخدمة في عدة تطبيقات كضغط بيانات تخطيط القلب الكهربائي(ECG) لتقليل حجمها مما يسهل عملية نقلها عبر الشبكة. بما ان بيانات الـ(ECG) الخاصة بالمرضى تنقل عبر الشبكة طول الوقت لذلك ظهرت الحاجة الى تقليل حجمها من اجل ضمان وصولها بالسرعة الممكنة لقاعدة البيانات. في هذه البحث نحن نهتم بطريقة الـ (LZW) التي هي واحدة من اهم واشهر طرق ضغط البيانات وقد اقترحنا بروتوكول لتحسين الطريقة التي تعتمدها خوارزمية الـ(LZW) في خزن المؤشرات الخاصة بالبيانات المضغوطة. البروتوكول المقترح يمكن ان يقلل حجم المؤشر لخوارزمية الـ(LZW). تم اعتماد خمس عينات اخذت من بنك المعلومات الخاص بـ(Physionet) لغرض اختبار البروتوكول المقترح. وقد اظهرت نتائج الاختبارت العملية ان البروتوكول المقترح يعطي نسبة ضغط افضل لبيانات الـ(ECG) مقارنة بطريقة الـ(LZW) الاصلية.

الكلمات المفتاحيه : LZW, ECG,ضغط البيانات, الضغط بدون فقدان.

1. Introduction

An ECG signals provide information about the electrical heart activity such as heart rate, rhythm and coprology which help the specialist to diagnostic any abnormality with the heart. Due to the vast ECG signal size and bounded of the network bandwidth, Compression technology is a practical solution while considering remote person identification and patient monitoring. Regular ECG signal implementation is restricted by the huge data storage requirement. Therefore, it can be clearly to notice that compressing the ECG signal is far more efficient when transmission and storage is contemplated [Iqbal, 2015].

All sorts of data stored on computer systems, or transferred via communication channels usually contain some sort of redundancy. Reducing the amount of redundancy would lead to a better storage utilization and faster data transfer. Different compression techniques for lossy and lossless data are available now. Due to the requirement of the ECG signal that have to preserve the exact signal for the diagnostic purpose, the signal before the compression should be exactly similar to the signal after the decompressed. For this only lossless techniques will be acceptable. However the lossy compression techniques will damage some details [Mathur, 2013; Salomon, 2005].

Conceptually, the lossless techniques are working through detecting and eliminating the redundant data that will be exist in the ECG signal. There are three reasons beyond using compression techniques for the ECG signals,

(i) To do more comparison and assessment for the ECG's as database, its size need to be growth.
(ii) Achieving the economic used of the wired or wireless network by transmission a compressed data.
(iii) Asking the doctor help for serious case form home or ambulance will be very useful [Mathur, 2013; Deorowicz, 2003].

LZW method is one of the important and famous data compression methods. It is a Dictionary-based coding technique that is particularly suitable for compressing digital data types. Typically, redundancy in digital data appears in the form of common digits, which repeat quite often. We propose a protocol for this method to perform a compressing for digital data especially ECG signal.

Dhar et.al. present an effective compression technique that depends on eliminating the high frequency and power line influence noise from the ECG signal by using the low pass filter and IIR notch filter respectively. Then strict lossless compression method has been applied on the enhance signal [Dhar *et. al.*, 2014]. Butta presents a lossless method that depend on a modified American standard code for information Interchange (ASCII) character encoding for ECG data compression. This method contain four parts which are algorithm comprising sign count; generation of array representing ECG sample's signs (+ve, -ve alternatively), adaptive amplification factor; and grouping method of ECG samples and a reverse process for ECG reconstruction [Singh *et. al.*, 2014].

In [Chang and Lin, 2010] an improved lossless algorithm for ECG signal has been presented by Gwo-Ching. The proposed method is depend on delta coding and optimal selective Huffman coding. Since the ECG signal rang is huge, delta code is used to reduce it. Also to improve the computation efficiency of canonical Huffman coding, optimal selective of Huffman code has been used.

Finally, Kumar presents a compression method for ECG signals that depends on beta wavelet using lossless encoding technique. The wavelet filter is used to reduce the compression distortion and to get more compression without any loss in the ECG signal, the run-length coding has been used [Kumar *et. al.*, 2013].

The proposed protocol focuses on how the indices of strings are stored in Dictionary-table. In original LZW, each index was stored as a block regardless of its exact needs to be represented in terms of bits. The proposed protocol stores each index in perfect size. We proved that the proposed protocol for LZW increases a compression ratio in compare to the original LZW method. The remainder of the paper is organized as follows: section (2) describes the old LZW method then section (3) shows the main problem in the old method. Section (4) presents the proposed method. Finally in section (5), the experimental results have been presented and concluded in section(6).

2. The LZW Compression Method

In the LZW algorithm the scenario has two processes, which are compressor and de-compressor. Compressor process reads the input byte by byte and adds them as a string of bytes (I). The compressor searches for string (I) in a dictionary. While the string (I) is found in the dictionary, the compressor reads new byte (a) and adds it to string (I), it will repeat the operation until it fails to find the string (Ia) in the

dictionary. When string (Ia) is not in the dictionary, the compressor produces a dictionary index that points to string (I) in dictionary [Perez et.al., 2005]. It stores string (Ia) as a dictionary entry in a next available place of dictionary and initialize the string (I) to byte (a). LZW algorithm is shown below: (λ denote the empty string, and <<a,b>> the concatenation of strings a and b)

```
/* set the table with all ASCII code strings */
Initialize dictionary-table with 1-symbol string for bytes from 0 to 255
Append λ to the dictionary;
Set a code to dictionary index of λ;   /* refer to empty string in table */
Do while more Input
    Input 1-byte (ch)                  /* one byte input */
 /* check when string stored in table at code value index with character of ch
if exist in dictionary */
    If concatenation of (code) and (ch) is in dictionary-table
            Set   code to dictionary index of << code, ch>>
    Otherwise
       Output (code);
       Append << code,ch>> to the dictionary;
       Set code to dictionary index of ch;
    End if
```

Moreover, in LZW compression the string of bytes is replaced with single code. The incoming bytes are just read without any analysis, and a new string of bytes is added to a table of strings. Compression is happen when a single code is replaced for a string of bytes [Nelson,1989]. The compression algorithm is based on a dictionary table that initially contains only the codeword for single bytes. For example, if the input is a 4-array numbers $\sum$ = {10, 20, 30, 40}, the initial table would be as shown in Table 1.

Table 1 the initial dictionary table

Input Bytes	10	20	30	40
Codeword	1	2	3	4

It is clear that the LZW compression algorithm increases the size of table by adding a new encountered string of bytes. It matches the input bytes to the already saved dictionary's strings of bytes. New strings are inserted to the table by adding one new byte to a previous string of bytes [Pfingsthorn and Liebald, 2003]. When a string is coded, its codeword is equal to its dictionary index. An example is a sequence 10,20,20,10,20,30,30,30,40,20,10. The input sequence is coded to 1 2 2 5 3 9 4 7 by using the compression steps shown in figure 1(a). The dictionary table for this example will show in Table 2.

In the decompressing state, the de-compressor process reads the codes and searches them in the dictionary table. The output in the decompression operation is the strings from the dictionary table instead of their indices as illustrated in figure 2.1(b) [Pfingsthorn, 2003, Shibata, 2003]. The dictionary table of the LZW algorithm is built during compression and decompression operation and it is not saved in the compressed data. Furthermore, the dictionary table is initiated with single byte code in compression and decompression operation. The other codes are added when bytes are input. Moreover, compression and decompression are work in similar way. In compression operation, the algorithm reads the bytes and produces the dictionary table. In decompression operation, the algorithm reads the code and reproduces the same dictionary table.

(a) Compression

Input data	10	20	20	10,20	30	30,30	40	20,10
Output codes	1	2	2	5	3	9	4	7

Strings add to dictionary table

10,20=5, 20,20=6, 20,10=7, 10,20,30=8, 30,30=9, 30,30,40=10, 40,20=11

(b) Decompression

Input codes	1	2	2	5	3	9	4	7
Output data	10	20	20	10,20	30	30,30	40	20,10

Strings add to dictionary table

10,20=5, 20,20=6, 20,10=7, 10,20,30=8, 30,30=9, 30,30,40=10, 40,20=11

Figure 1 Example for compression and decompression with LZW

Table 2 the dictionary-table for sequence 10,20,20,10,20,30,30,30,40,20,10

Symbols	10	20	30	40	10,20	20,20	20,10	10,20,30	30,30	30,30,40	40,20
Codeword	1	2	3	4	5	6	7	8	9	10	11

The decompression algorithm is as following: (λ denote the empty string, and <<a,b>> the concatenation of strings a and b)

```
Initialize dictionary-table with 1-symbol string for bytes from 0 to 255
    Input (old_code);
    Output (old_code);
  Set ch to (old_code);
   While more code input
       Input (new_code)
        If new_code is not in dictionary_table then
            Set ST to string of (old_code)
```

```
    Set ST to << ST,ch>>
  Otherwise
    Set ST to string of (new_code)
  endif
 Output (ST)
   Set (ch) to first character of (ST)
  Append <<old_code,ch>> to dictionary-table
  Set (old_code) to (new_code)
```

3. The main problem in the original LZW

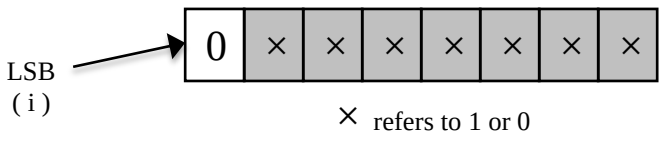
In the LZW compression, the size of produced codes (indices) that is the real compressed data is not equivalent. Furthermore, the index could be one or more byte in terms of size. Each index is saved in the produced compress data as a fixed size block (not with its specific size). They are stored as blocks of data that are with the same size. Some programmer makes the block size two bytes and that may not be enough for large size index. Most other makes the block size three or four bytes. However, some indices need one, two, three or more bytes to be saved. This way leads to loss some bytes for index saving. When the index is stored in its exact size, it would be difficult to distinguish among them, at the decompressor side. In this paper, we develop a protocol for the LZW algorithm where it becomes possible to store each index in its exact size.

4. Proposed Protocol

The proposed protocol classifies the codes (indices) that are received from LZW scheme according to its length into classes (A, B, C…). The length of index belong to class A is 1-byte, class B is 2-byte and so on as shown in table 4-1. In the compressing process, the Left Most Significant Bit (MSB) of index is set to zero for class A as illustrated in figure 3. The other seven bits could be zero or one. In the decompressing process, the index of class A is distinguished when its LSB is zero.

Table 3 index classifications of LZW scheme

Index class	Index length/byte
A	1
B	2
C	3
D	4
.	.
.	.
.	.



× refers to 1 or 0

LSB: Left Significant Bit

Figure 3 LZW Index of class A

The index of class B has 2-byte size. In the compressing process, the MSB (i) is set to one, and bit (i-1) is set to zero as illustrated in figure 4. In the decompressing process, the index of class B is distinguished when the MSB is one followed by zero. For index of type class C, the MSB must be one followed by one then zero. In general, the length of index can be computed by a set of ones followed by zero as pointed in equation 1.

$$Index\_length = 9 - i \qquad Eq.\ 1$$

where *(i) location of bit set to zero*
bit (i) = 0 *index-length : no. of byte*
bit (j) = 1 , 9 > j > i

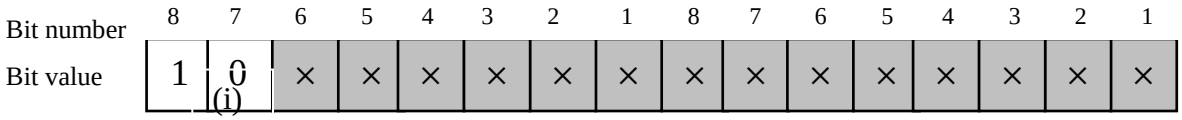


Figure 4 LZW index class B (2-byte)

The using of the proposed protocol for LZW algorithm made the index of codeword should belong to one of the classes. It could of class (A, B, C…) as illustrated in table 4. Moreover, each index of codeword is classified by computing the position of bit (i) as pointed in equation 1.

Table 4 LZW index

Index Class	First Byte							
A	0	×	×	×	×	×	×	×
B	1	0	×	×	×	×	×	×
C	1	1	0	×	×	×	×	×
D	1	1	1	0	×	×	×	×
E	1	1	1	1	0	×	×	×

the symbol × equals to zero or one

The proposed protocol for LZW algorithm would change the shape of the original algorithm as following:

Variables:

λ: empty string
ch: 1-byte
ST: string of bytes
code, old_code and new_code: integer value
dictionary-table: dynamic array

A-Compression

```
Initialize dictionary-table with 1-symbol string for bytes from 0 to 255
Append λ to the dictionary
Set (code) to dictionary index of λ
Do while more input
    Input (ch)
    If << code,ch>> is in the dictionary then
        Set (code) to dictionary index of << code,ch>>
    Otherwise
        Classify (code) according to its length
        Output (code)
        Append << code,ch>> to the dictionary
        Set (code) to dictionary index of ch
    End if
```

B-Decompression

```
Initialize dictionary-table with 1-symbol string for bytes from 0 to 255
Input (old_code);
Output (old_code);
Set (ch) to (old_code);
While more input
        Input (new_code)
        Classify (new_code)
          If (new_code) is not in dictionary
                Set (ST) to string of (old_code)
                Concatenate (ST) with (ch)
            Otherwise
              Set (ST) to string of (new_code)
           End if
          Output (ST);
          Set (ch) to first character of (ST);
          Add <<old_code,ch>> to dictionary-table
          Set (old_code) to new_code;
```

5. Evaluation

The data samples of human ECG were used to compare the mean of compression ratio between the old and the new methods. Compression ratio CR is calculated by dividing the uncompressed size US by the compressed size CS. Five

sample data groups provided by Physionet were used for evaluation [Goldberger, 2010]. Explanation of the data groups is as follows:
OB-1 database contain a set of recordings of fetal scalp Electrograms and uterine muscular activity, with beat-by-beat annotations of the fetal ECG, to support studies of fetal heart rate variability. ANSI/AAMI EC13 Test Waveforms. These 10 short recordings are specified by the current American National Standard for testing various devices that measure heart rate where each recording contains one ECG signal. Paroxysmal or sustained atrial fibrillation database includes long-term ECG recordings of subjects with paroxysmal or sustained fibrillation (AF). MGH/MF Waveform Database. This is a collection of 250 recordings of 3-lead ECGs, ABP, PAP, CVP, respiration, and airway CO2 signals from patients in critical care units; some recordings include intra-cranial, left atrial, ventricular and intra-aortic pressure waveforms. Finally, Prediction Challenge Database (PAF), it consists of 100 record sets, each including a pair of 30-minute excerpts from a long-term ECG recording.

In each sample data group, 10 samples were used. Each sample group contains ECG data of fixed period. The results are shown in Table 5

Table 5

Database	Old-LZW	New-LZW
OB-1	0.693945	0.594902
ANSI/AAMI EC13	0.775	0.604864
AF-long	0.600977	0.51388
MGH/MF	0.685352	0.618262
PAF	0.867773	0.758926

The results in Table 1 show that the presented technique has the highest compression ratio than the old one. Table 6 shows the mean compression ratio of the new and old compression algorithm, which are 0.68, 0.72 respectively.

Table 6

Old-LZW	0.72
NEW-LZW	0.61

Also, the proposed research shows that it is high in objectivity because five types of databases are used instead of just one or two, like in other related researches.

6. Conclusions

The LZW compression algorithm is adopted for many applications. The ECG data could be compressed by the LZW algorithm where the ration of data compression depends on the nature of data or the redundant of bytes. The LZW compresses the ECG data as blocks of codeword (index) and this leads to decrease the compression ratio.

Moreover the proposed protocol for ECG data compression improves the LZW algorithm and enables the LZW to compress the codeword in perfect size. The proposed protocol deals with dynamic block rather than fixed block codeword. In compare with the original LZW algorithm, the experimental result of ECG data compression shows that the proposed protocol for LZW scheme increases the compression rate.

References

-Brisaboa N.R., Fariña A., Navarro G., and José R., "Simple, Fast, and Efficient Natural Language Adaptive Compression", 11th International Conference, SPIRE 2004, Padova, Italy, October 5-8, 2004, Springer Science + Business Media, Inc, 2005.

- Chang G.C, Lin Y.D. "*An Efficient Lossless ECG Compression Method Using Delta Coding and Optimal Selective Huffman Coding*" IFMBE proceedings 2010, Volume 31, Part 6, 1327-1330

-Deorowicz S., "*Universal lossless data compression algorithms*", Doctor of Philosophy Dissertation, Supervisor: Prof. dr hab. inz. Zbigniew J. Czech, Silesian University of Technology, Faculty of Automatic Control, Electronics and Computer Science, Institute of Computer Science, 2003.

- Dhar, S.; Mukhopadhyay, S.K.; Mitra, S.; Baig, M.M.; Mitra, M., "Noise reduction and lossless ECG encoding," in *Control, Instrumentation, Energy and Communication (CIEC)*, 2014 International Conference on , vol., no., pp.210-213, Jan. 31 2014-Feb. 2 2014.

- Goldberger, A. L., Amaral, L. A. N., Glass, L., Hausdorff, J. M.,Ivanov, P. C., Mark, R. G., Mietus, J. E., Moody, G. B., Peng, C.-K.,and Stanley, H. E., PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals. Circulation 101(23):e215–e220, 2010. doi:10.1161/01.CIR.101.23.e215.

- Iqbal F. and Sidek K. A., "*Compressed ECG Biometric Using Cardioid Graph Based Feature Extraction*", ARPN Journal of Engineering and Applied Sciences, vol. 10, no. 22, pp. 17219-17224, 2015.

- Kumar R., Kumar A. and Pandey R.K., "*Beta wavelet based ECG signal compression using lossless encoding with modified thresholding*" Computer and Electrical Engineering, vol. 39, no. 1, pp. 130-140, 2013

- Mathur M. K. , Garg A. K. , Upadhayay M., " *Application of LZW Technique for ECG Data Compression",* International Journal of Advances in Computer Networks and its Security, 2013.

- Nelson M., ''*LZW Data Compression*'', 1989, available at: http:/www.dogma.net/markn/articles/lzw/lzw.htm

-Pérez A.C., Uribe C.F., and Navarro G. "*Approximate Searching on Compressed Text*", Proceedings of the 15th International Conference on Electronics, Communications and Computers (CONIELECOMP 2005) 0-7695-2283-1/05 IEEE 2005.

-Pfingsthorn M. and Liebald B., "*On the LZW compression algorithm*", November 15th , 2003, available at: http:/www.catenary.com/appnotes/lzwcomp.html

-Singh B., Sharma D., Singh M. and Singh D. 2014. "*An improved ASCII character encoding method for lossless ECG compression*". Advances in Biomedical Science and Eng., 1(2), 1-11.

- Salomon D., ''*A Guide to Data Compression Methods*'', Springer, 2002.

-Shibata Y., Kida T., Fukamachi S., Takeda M., Shinohara A., Shinohara T., and Arikawa S., "*Speeding Up Pattern Matching by Text Compression*", CIAC2000, LNCS 1767, pp. 306-315, Springer-Verlag Berlin Heidelberg 2000.