Improved LZW Algorithm of Lossless Data Compression for WSN

Zhou Yan-li, Fan Xiao-ping, Liu Shao-qiang, Xiong Zhe-yuan
School of Information Science and Engineering, Central South University, Changsha, Hunan, China
zhou-yanli2007@163.com

Abstract—Wireless Sensor Network (WSN) nodes are powered by limited battery and communication consumes most of the energy. Therefore, it's not appropriate to transmit data directly in the networks while compressing data before sending is an effective method to save energy for nodes. According to the need for fully recovering the compressed numerical data, based on the LZW (Lempel-Ziv-Welch) algorithm, this paper proposes an improved lossless data compression algorithm for WSN nodes. In this algorithm, the span of data to be compressed is reduced by calculating increment between two adjacent data of sample sequence. The increment is replaced by a single character, where \(1 \sim 25\) is orderly replaced by \(A \sim Y\), \(-1 \sim -25\) by \(a \sim y\), 0 by \(Z\), and the data out of the span is marked by character \(z\), so as to save memory space and improve data duplication possibility. Taking into account the different real-time requirements on data compressing, two specific algorithms for real-time transmission and latency tolerant transmission are given. Comparing test between the improved LZW algorithm and the original is made, and the results of testing data compression show that the two improved LZW algorithms have remarkable superiority in reducing dictionary length and increasing the compression ratio.

Keywords—lossless compressing data; wireless sensor networks; improved LZW algorithm; increment; resource limited system

I. INTRODUCTION

Data communication is the most energy-consuming task for sensor nodes in WSN. As the energy supply, processing ability, storage capacity and communication bandwidth of nodes are limited, it is not appropriate to transfer original measured data directly generally. Fortunately, trading computation for communication may save energy. One research shows that the energy consumption for executing 3000 thousands instructions is equal to the energy consumption for transmitting one thousand bit over a distance of 100 m by radio [1]. Therefore, compressing data before sending is an effective way to make good use of nodes’ limited resources and reduce energy consumption of nodes and networks [2].

Data compression includes lossy compression and lossless compression. The main factor that influences data compression for WSN is the limited resources of sensor nodes. For this reason, data compression algorithms that specially designed for WSN must focus on low computation complexity and high energy efficiency [3].

In the past few years, there have been a lot of researches about data compression for WSN. Wavelet-based data compression is one of the hot topics currently [4][5], but they are lossy compression. Important measured data needs to be stored in the locality without any loss, so as to prepare for inquiring. For real-time monitoring WSN, the raw data may need to be transmitted to a higher level node or a sink node. Thus, energy-efficient lossless compression methods are needed to meet the local storage and wireless transmission requirements of WSN.

There are many lossless data compression algorithms can be applied to WSN, such as Huffman coding, LZW coding and run-length coding [6]. Among these coding schemes, LZW algorithm is the remarkable one. It generates different dictionaries according to the different compressed contents, and the dictionary does not need to be sent with code simultaneously but gradually established at the decoder. It has the advantage of being adaptive and does not assume any priori knowledge of the source properties. The algorithm is easy to implement and its performance in latency, memory, compression effect are all suitable for WSN, but its coding efficiency still needs to be improved. According to different latency requirements of data transmission in wireless sensor network, this paper proposes two algorithms specific for real-time transmission and latency tolerant transmissions. Implement program is designed based on ultra-low power microcontroller development environment. Pseudo-periodic pulse wave measurement data is selected to test the algorithm and the performance comparison between proposed algorithms and original.

II. LZW ALGORITHM

LZW compression is a dictionary lookup-based algorithm [7], namely any string’s prefix string is also in the dictionary. If character string \(ω\) consisted by character string \(ω\) and a single character \(K\) is in the dictionary, and then \(ω\) is also in the dictionary. When the character string appears again while encoding, it can be replaced by a figure which represents it and output the figure. The dictionary here mentioned is not built in advance, but dynamically built based on raw data. The raw dictionary is recovered from the decoded data while decoding. The dictionary can be correctly built during the course of compression and decompression, and discarded after compressing or decompressing process has completed. In short, a dictionary, which contains non-repetition sub-string of input stream, is adaptively built in LZW algorithm. And then, each sub-string
is mapped into an independent output codeword. Consequently, it achieves the purpose of data compression. There are three steps should be followed when compressing text files with LZW compression algorithm:

a) Put all single characters into the dictionary.

b) Read the first character and evaluate it to the prefix string.

c) Read the next character K, if it’s the end of file, then output the dictionary and quit. Otherwise, if $\omega K$ is in the dictionary, $\omega = \omega K$, repeat (c); else add $\omega K$ to the dictionary, $\omega = K$. If $\omega$ is contained of a single character then output it, if $\omega$ is contained of multi characters, output the address corresponding with string. Repeat (c).

III. THE IMPROVEMENT AND IMPLEMENTATION OF LZW ALGORITHM

A. Limitations of LZW algorithm

Although the logic of LZW algorithm is simple and easy to implement, there are a few limitations or deficiencies on the encoding mode:

a) All single characters need to be put into the dictionary at first, but they don’t need involving in the encoding and decoding process, so the LZW algorithm has the disadvantage of space redundancy.

b) There are so few words in the dictionary in the preliminary stage that dictionary contributes little to compression effect, so it is not suitable for a small amount of data. On the other hand, for limited storage capacity the dictionary may overflow when the data is too large and its efficiency will also be limited. Thus, LZW compression algorithm is not suitable for a large amount of data.

c) The length of dictionary’s substring is not restrained, while longer substrings generally don’t appear frequently. Especially for the low-repetition-ratio continuous numerical data, it increases the size of the dictionary and contributes little to the improvement of compression ratio.

d) LZW algorithm is only suited for text files.

B. Major improvements of LZW algorithm

For typical sensor nodes, which are embedded with ultra-low power microcontroller, such as MSP430F series whose storage and calculating power are limited, the limitation mentioned above will increase the burden of nodes’ storage or calculating. Therefore, this paper presents the following improvements in the encoding process:

a) All single characters will no longer be put into the dictionary at first, which can reduce the dictionary size and also be correctly decoded.

b) Select appropriate dictionary capacity. As nodes are limited in memory space, it is suitable to store the dictionary with address space of two bytes which is sufficient for compression in nodes.

c) Limit the length of substring in the dictionary, because appropriate length can both save memory space and maintain high efficiency. The length is not determined, but can only be got through testing. For numerical data, we can adopt difference method to reduce the data range, which is beneficial to improve data duplication possibility and reduce the size of dictionary.

d) Carry out preprocessing for different situations to convert other non-document file into text file.

C. Major procedures of improved LZW algorithm

LZW algorithm is based on dictionary and the compression effect depends on the degree of repetition between latter data and former data, the higher the better. Improved algorithms proposed in this paper aim at continuous numerical data with a base value drift characteristic. In order to achieve better compression effect, the increment between two adjacent data should be calculated at first. As the LZW algorithm only suited for characters, the increment also needs to be converted into character.

Suppose that $\beta_i$ is a sample sequence, $\alpha_i$ is a difference value sequence obtained by performing difference value between two adjacent data, that is $\alpha_i = \beta_i - \beta_{i-1}$.

Suppose that $\gamma_i$ is a sequence preparing for encoding after character replacing, single character sequence $\delta_i = \{A, B, \ldots, Y\}$, $u_i = \{a, b, \ldots, y\}$.

If $(\alpha_i = 0)$ $\gamma_i = Z$;

Else

If $(\alpha_i > 0 \&\& \alpha_i \leq 25)$ $\gamma_i = \delta_{u_i-1}$;

Else

If $(\alpha_i < 0 \&\& \alpha_i \geq -25)$ $\gamma_i = e_{u_i-1}$;

Else $\gamma_i = z$;

Raw integral data occupies two bytes while character data only occupies one byte after pretreatment, which can save memory space.

Depending on the nodes’ different requirements of data transmission latency, two improved algorithms are proposed. The main difference between them is the processing mode of increment single character $\gamma$ represents. When the node tolerates data transmission latency, the increments can be processed in batch. All increments single character $\gamma$ represents are outputted orderly and $z$ will be involved in coding which will be orderly stored in the decoding process. As data can only be transmitted in binary system, character and address must be distinguished.

Suppose that single character $\eta_i \in \{A, Z\} \cup \{a, z\}$, that is $[01000001,01011010] \cup [01100001,01111010]$ expressed in binary system. Suppose that $\theta_i$ is an output data sequence, as the highest digit of $\eta_i$ is 0, take the highest bit of binary digit into negation, namely $\theta_i = \eta_i \mid 10000000$, in order to ensure the highest bit is 1.

For two-byte address, the highest digit of binary is set to zero in order to ensure the maximum dictionary capacity is 32768, if the dictionary overflows then re-encoding so as to meet the requirements of nodes’ limited memory space. If
the highest digit is zero, the following two bytes expresses address. Address and character are distinguished in this way.

When the nodes has a real-time transmission requirement, transmitting data that z represents in batch will cause greater delay. Thus, the method of processing data in sequence is adopted, that is to say, obtaining increment, character replacing, coding and sending will be performed in turn after reading a value. While meeting the increments that exceed the range, the character z, difference value and position information z represents will be outputted orderly, z will not be involved in encoding. So long as the single character is judged to be z in decoding process, the following four bytes must be increment and location information.

1) Latency tolerance algorithm

When the sensor node has low real-time requirements, the following steps will be taken to implement the algorithm:

a) Read the first data of array β and output it.

b) Calculate the increments according to the formula \( \alpha_i = \beta_{i+1} - \beta_i \) until the end of the file.

c) Replace values in array \( \alpha \) with characters. If \( \alpha_i \leq 25 \), replace values 1~25 with characters A~Y, -1 ~ -25 with characters a~y, 0 with character Z and store them in array γ. Otherwise replace \( \alpha_i \) with z and output \( \alpha_i \).

d) Read the first character and evaluate it to the prefix string o.

e) Read the next character γ, if it’s the end of file, then quit. Otherwise combine o with γ and judge whether the length of oγ is longer than you set. If it’s longer, output it, else, \( o = oγ \), repeat (e).

f) Repeat (d) until the end of the file, then quit. Otherwise, \( o = oγ \), output it, and perform (e).

In this case, the data can be processed in batch. Therefore, the algorithm can be separated into pre-processing module and encoding module. The flow charts are shown in Figure 1 and Figure 2.

2) Real-time algorithm

When the sensor node has high real-time requirements, the following steps will be taken to implement the algorithm:

a) Read the first data in \( \beta_i \) and output it.

b) Read the next data \( \beta_{i+1} \) and calculate the increments according to the formula \( \alpha_i = \beta_{i+1} - \beta_i \) if \( i = 1 \), output \( \beta_i \). If \( \alpha_i \leq 25 \), replace values 1~25 with characters A~Y, -1 ~ -25 with characters a~y, 0 with character Z and store them in array γ. Otherwise output character z, \( \beta_{i+1} \) and i, i++, repeat (b).

c) If \( i = 1 \), evaluate character to the prefix string o. Otherwise, judge whether oγ is in the dictionary. If not, perform (d), else \( o = oγ \), i++, perform (b).

d) Put oγ into dictionary and judge whether o is consisted of single character or multi characters. For the former case, output it. For the later case, inquiry the address of o in the dictionary and output the address, \( o = γ \), i++,

As the data is processed in sequence, pre-processing and encoding processes can be performed in sequence. The flow chart is shown in Figure 3.
IV. THE EXPERIMENTAL RESULTS AND ANALYSIS

To verify the effectiveness of the two algorithms, we select MSP430F1611 and MSP430F1612 ultra-low power microcontrollers used by common nodes and adopt IAR 430 development environment for the compression test of sample data. Data is stored by array. Both stack and memory are configured according to the actual size of chip’s data storage capacity.

In this paper, pulse wave signal data are used as sample data to test the algorithms. The data is continuous but with a few dramatic changes. The absolute value of increments are almost lower than 12. If the increments between [-25, 25], they are all replaced by characters, it may result in space redundancy, low data duplication possibility and large dictionary. Therefore, the difference range in improved algorithms is taken between -12~12, of which 1 ~ 12 orderly replaced by A ~ L, -1 ~ -12 orderly replaced by N~Y, 0 replaced by Z, the others are all replaced by M. Different sizes of pseudo-periodic pulse wave measured data are selected and the signal waveform is shown in Figure 4.

Test results are shown in Table 1 among which compression ratio refers to the ratio between the reduction of data size and the raw data size. Improved LZW algorithm I means algorithm with latency tolerance, improved LZW algorithm II means real-time algorithm.

We can summarize three conclusions after comparing the test results of three algorithms:

a) For the same algorithm, compression ratio gradually increases with the increasing of data size. However, the growth range gets smaller and smaller, which indicates that the compression ratio will not always roll up but infinitely close to a certain value. The value depends on the compressed data, different data results in different value.

b) Regardless of the amount of data size, both the dictionary and the output size of the two improved algorithms are much smaller than that of the original algorithm. Meanwhile, compression ratio is much higher which means that the improved algorithms indeed have a good compression performance and save a lot of memory.

c) Comparing the two improved algorithms, we can conclude that the dictionary size of improved LZW algorithm I, in which single character M is involved in encoding, is slightly larger than algorithm II, while the data repetition ratio is higher which causes smaller output and higher compression ratio. In algorithm II, character M and the difference value it represents as well as the location of the raw value must be involved in transmission, all of which occupy 5 bytes, while the difference value represented by M in algorithm I is only involved in transmission which occupies two bytes. The ratio of algorithm II is slightly smaller because its output is greater than the output of algorithm I. But algorithm II occupies smaller space and has real-time performance, both have their own advantages.

![Figure 4. Pulse Waveform](image-url)

### TABLE I. TEST RESULTS OF THREE ALGORITHMS

<table>
<thead>
<tr>
<th>Raw data size (bytes)</th>
<th>LZW algorithm</th>
<th>Improved LZW algorithm I</th>
<th>Improved LZW algorithm II</th>
</tr>
</thead>
<tbody>
<tr>
<td>4800</td>
<td>6513</td>
<td>3389</td>
<td>0.296</td>
</tr>
<tr>
<td>6000</td>
<td>7912</td>
<td>4036</td>
<td>0.327</td>
</tr>
<tr>
<td>7200</td>
<td>9206</td>
<td>4621</td>
<td>0.358</td>
</tr>
<tr>
<td>8400</td>
<td>10455</td>
<td>5196</td>
<td>0.381</td>
</tr>
<tr>
<td>9600</td>
<td>11527</td>
<td>5677</td>
<td>0.409</td>
</tr>
</tbody>
</table>
The energy consumed by access to the on-chip memory is significantly lower than the energy consumed by access to off-chip memory in microcontroller [8], so it is better to use on-chip memory in order to reduce energy consumed by calculating. The two algorithms can be implemented only using the registers and on-chip memory through appropriate allocation of resources and variable multiplexing technique, which is conducive to energy conservation.

V. CONCLUSION

The sensor nodes are limited in memory space and resource, and the LZW algorithm has low efficiency in time and space when encoding, taking into account sensor nodes' different degrees of real-time requirements, two improved algorithms are proposed. The results of compression test on sample data show that these two algorithms can significantly improve the compression ratio and reduce the size of the dictionary as well as energy consumption in data transmission. Both of them reduce the data amount of storage and transmission which contributes to the energy saving.

ACKNOWLEDGMENT

This work was supported by a grant from the National Natural Science Foundation of China (No.60776834 and No.60870010).

REFERENCES


