COMPARATIVE STUDY ON DATA COMPRESSION ALGORITHM FOR WIRELESS SENSOR NETWORK

A. Jeevarathinam[1], *Dr. S. Manju Priya*[2]

Abstract

Data compression techniques are emerging trends in sensor networks. It is an art of compact information. It is used to decrease the amount of data transmitted by source node.

It contains Data generating and sink node. Data generating nodes acts as the encoding instruction and sink node act as decoding instruction. In this paper, a loss less dictionary based data compression technique has been proposed which is based on the optimality of LZW code

Keywords : WSN, Data Compression, LZ77, LZ78, LZW;

I. Introduction

A WSN is a set of nodes structured to form a network. Wireless Sensor Networks are made up of small sensor nodes. It is used for monitoring and sensing information [7]. No a days WSN status and performances are arising then automatically the limitations of the network lifetime also arising. It is observed that the energy utilized in transmission is more than processing the data in WSN. So, a lot of approaches are planned to attempt the problem of power utilization, used while transmission. Data compression is one of the approaches by using data to be transmitted and recovery of it at sink node in WSN [5].

[1]Research Scholar, Department of Computer Science, Karpagam Academy of Higher Education, Karpagam University
[2]Associate Professor, Department of Computer Science, Karpagam Academy of Higer Education, Karpagam University

The major task of data compression is to remove the unnecessary data. By compressing the data, the size is reduced and less bandwidth is required for transmitting data [6]. It can be classified as lossless, lossy and unrecoverable compression Lossless data compression recreates the exact original data from the compressed data while lossy data compression cannot regenerate the perfect original data from the compressed data. The Unrecoverable compression means that the compression operation is irreversible; there is no decompression operation [3].

There are plenty of data compression algorithms which are coupled for WSN. It can be categorized as static, adaptive or dynamic, and hybrid. The Static encoding requires 2 passes. The first pass is to determine the frequencies and mapping techniques [9]. The Second Pass to encode the data. Huffman Coding is the best example of Static encoding. One scan of the message is essential for the adaptive techniques. Some of the adaptive techniques are Lempel Ziv 77, Lempel Ziv 78, Lempel Ziv Welch, and Adaptive Huffman Coding [4].

One of the lossless data compression algorithms is Lempel Ziv Welch (LZW). In the late 1970's better methodologies where developed for Huffman coding using Data compression. In 1977, Abraham Lempel and Jacob Ziv proposed different method of encoding algorithms. Lempel-Ziv techniques consist of two different algorithms, LZ77 and LZ78.Due to patents, LZ77 and LZ78 led to many variants.[2]

II LZ77 COMPRESSION ALGORITHM

It is a Sliding Window method for encoding the characters. In this technique, the dictionary work as a segment of the previously encoded series. The encoder examines the input series through a sliding window. It consists of two parts one is Search buffer and another is Look-ahead buffer [9]. A search buffer contains a part of the recently encoded series and a look-ahead buffer contains the next portion of the series to be encoded. LZ77 was better but slower, but the gzip version is almost as fast as any LZ78 [1]. The LZ77 algorithm describes the process of encoding the characters using LZ77 is as follows [1].

Algorithm : LZ77

```
While (LAB #=0)
 {
Get a pointer (P, L) to longest match;
if (L > 0)
{
Result (P, Longest match length, next symbol);
Shift the window by (length+1) positions along;
}
Else
{
Result (0, 0, first symbol in the LAB);
Shift the window by 1 character along;
}
 }
```

LAB - look-Ahead Buffer

P- Pointer

L-Length

Example :

J	**K**	**L**	**J**	**M**	**J**	**N**	**J**	**K**	**L**	**J**

	Address	**Length**	**Deviating Symbol**
JKLJMJNJKLJ	**0**	**0**	'J'
J KLJMJNJKLJ	**0**	**0**	'K'
JK LJMJNJKLJ	**0**	**0**	'L'
JKL JMJNJKLJ	**3**	**1**	'M'
JKLJM JNJKLJ	**2**	**1**	'N'
JKLJMJN JKLJ	**7**	**4**	' '

If there is no matching series is exist from the past content, the series length 0,the address 0 and the new symbol will be coded in to the dictionary. Because all byte series is extended by the first symbol differing from the past contents, the list of already used symbols will constantly raise. No additional coding scheme is necessary. This allows a simple execution with least requirements to the encoder and decoder[10].

Advantages

This technique searches the data of thousands of bytes long for search buffer and tens of byte long for look-Ahead Buffer. The encoding method is time consuming. It can be used to find matched pattern using hug amount of comparisons. It causes problem, while decompressing on another machine because it doesn't have its external dictionary. In this algorithm when there is no match of any strings, it encodes that string as a length and offset, which will obtain extra space and this needless step increases the time period of the algorithm

Disadvantages

This technique attempts to work on past data. It is slow in compressing the data. The result format of LZ77 is triple of P, L, S, Where P=Position of the longest match, L=length of the match, S=next symbol to be encoded. It is preferred using backward pointers. It is mainly due to the limits of data. This problem is most evident when compressing a long periodic sequence in which the period exceeds the length of the search buffer.

III. LZ78 Compression Algorithm

It is a Dictionary Based encoding character which is to inserted one- or multi-character, non-overlapping, discrete patterns of the message to be encoded in a Dictionary. It is a easy method which haves a window as a dictionary. When compressing the data we can use {I,S} for access the dictionary. The 'I' is the index position of the dictionary and 'S' is the String that is after the match. [8]

Example :

JKKLKKJJKLMMKLLJJKJJKL

Alphabet A= {J,K,L,M} is using for compressing the sequence.

At First, the Dictionary is empty. Then we can add one by one character or string in to the dictionary. At the beginning, we can add first character 'J' in to the dictionary. l is the index position and J is the Character which we have to insert. So the result will be {0, J}. Then next character K is inserted in to the dictionary at the index position 2.So the result will be in the form of {0,K}.The next character K is in the Dictionary and KL is not in the Dictionary; so insert it at the position of 3.So the result become{2,L}.Similarly the remaining characters are inserted in to the dictionary. At the end of the strings are shown in the table [1]. The LZ78 algorithm describes the process of encoding the characters

Algorithm : LZ78

```
Word: = Empty;
While (INPUT)
{
Key: = next String from input;
If (word Key exists in the dictionary)
{
word : = wordKey;
}
Else
{
Result (index (word), Key);
Add word Key to the dictionary;
Word: = Empty;
}
}
```

Example : Table I Dictionary

INPUT STRING :

JKKLKKJJKLMMKLLJJKJJKL

Index	1	2	3	4	5
String	J	K	KL	KK	JJ
Result	{0,J}	{0,K}	{2,L}	{2,K}	{1,J}

Index	6	7	8	9	10
String	KLM	M	KLL	JJK	JJKL
Result	{3,M}	{0,M}	{3,L}	{5,K}	{9,L}

Advantages of LZ78

This technique helps to proceed for future data. It is faster than LZ77 and the result format of LZ78 is pair of I, S. Where I is index and S is a String. It is preferred using a real dictionary. LZ78 technique overcome the problem of the period exceeds the length of the search buffer when compressing a long periodic sequence; it indeed stores the patterns within a dictionary as tokens.

Disadvantages of LZ78

No more entries are added when the dictionary is full. Last matching index is the output of the algorithm, when the closing stage of the input stream is reached. Reverse order of the string are stored in the dictionary.

IV. LZW Compression Algorithm

LZW algorithm is the most popular algorithm. It is proposed as a variant of LZ78 algorithms, where the compressor never outputs a character, it always outputs a code. To perform this, a major modification in LZW is to be performed in the dictionary with all possible symbols that can occur. It is just like a greedy approach which divides text into substrings. Instead of having a double, $<i, s>$, the LZW uses a techniques to remove the need for the second field in the double.

LZW uses fixed-length codewords to represent variable-length strings of symbols/characters that commonly occur together, e.g., words in English text. LZW compression and decompression make up the similar dictionary as getting the data. It occupies more and more frequent entries into a dictionary. In this algorithm uses the dictionary of 256 character sets and reads 8 bits of data at a time. Number is the input data that represent its index in the dictionary. If a new sub string comes, it could be inserted into the dictionary. If a sub string has already seen, it immediately reads in a new character and joins it with the existing string to construct a new substring. Subsequently LZW visit often a substring, it will be encoded using a single number. Frequently the greatest amount of entries (2048) is defined for the dictionary, so that the method doesn't run away with memory. Hence**,** the codes which are taking place of the substrings in this example are 12 bits long (211 =2048).

ALGORITHM : LZW Compressions

```
BEGIN
String = next input character;
While # EOF
{
Char = next input character;
if string + char available in the dictionary
String = string + char;
Else
{
Result the code for string;
Add string + char to the dictionary with a new code;
String = char;
}
}
Result the code for string;
END
```

Example : Table II: Dictionary Consider a string table containing only 6 characters, with codes as follows :

Index	1	2	3	4	5	6
String	J	K	L	M	JK	KK

In case, the input string is **JKKLKKJJKLMMKLLJJKJJKL**, the LZW compression algorithm works as follows

Index	7	8	9	10	11	12
Input String	KL	LK	KKJ	JJ	JKL	LM
Result	2	3	6	1	5	3

Index	13	14	15	16	17	18	19
Input String	MM	MK	KLL	LJ	JJK	KJ	JJKL
Result	4	4	7	3	10	2	17

The output codes are: 2 3 6 1 5 3 4 4 7 3 10 2 17. Instead of sending 19 characters, only 9 codes need to be sent (compression ratio = 19/13 = 1.46).

ALGORITHM : LZW Decompression

```
BEGIN
String = empty;
While # EOF
{
Key = next input character;
Empty= dictionary entry for k;
Result entry,
if (String != empty)
Add string + char[0] to the dictionary with a new code;
String = entry;
}
END
```

Example 7.3 : LZW decompression for string **JKKLKKJJKLMMKLLJJKJJKL**

Input codes to the decoder are 2 3 6 1 5 3 4 4 7 3 10 2 17. The initial string table is the same to what is used by the encoder. The LZW decompression algorithm then works as follows :

Table III : Dictionary

Index	7	8	9	10	11	12	13
Input Code	2	3	6	1	5	3	4
Result	KL	LK	KKJ	JJ	JKL	LM	MM

Index	14	15	16	17	18	19
Input Code	4	7	3	10	2	17
Result	MK	KLL	LJ	JJK	KJ	JJKL

Apparently, the output string is JKJKKJKLJKJKKJ ", a truly lossless result!

Advantages of LZW :

LZW is loss less compression technique. It is very fast to decode the data and also very simple to implement and analyze the incoming data.

Disadvantages of LZW :

LZW is simple method but implementation of algorithm is very complex, because of managing string table. The volume of storage required is undefined as it depends on the sum of all strings. A problem occurs while searching the data, if a new character is read in at every time, the algorithm has to search for the new string formed by string+character. Whenever a new character comes, it could be searched in to the string table. If a particular character or string is not found then a new character has to be added to the string table. It causes two problems: The string table gets enlarged very fast. If the string lengths average is low as three or four character each, then the overhead of storing the variable length string and its code could reach seven or eight bytes per code.

V. Comparison between LZ77, LZ78 and LZW

Comparison between LZ77, LZ78 AND LZW algorithms for WSN is described in the following table 1V

Table 1V : Comparison between LZ77, LZ78 AND LZW Algorithms

LZ77	LZ78	LZW
❖ It works on past information for encoding and decoding	❖ It works on future information for encoding and decoding	❖ It works on future information for encoding and decoding
❖ It is slow	❖ It is fast	❖ It is very faster than LZ77 and 78
❖ The result format of LZ77 is triple of P, L, S.	❖ The result format of LZ78 is pair of I,S	❖ The result format of LZ78 is pair of I,S and code words
❖ It is preferred using backward pointers	❖ It is preferred using a real dictionary, it indeed stores the patterns within a dictionary as tokens	❖ LZw compression and decompression make up the same dictionary

VI. Conclusion

This paper has taken up Lempel Ziv algorithms to observe the performance in data compression. Comparison of LZ77, LZ78 and LZW techniques are described. The dictionary based encoding is shown up in the comparison table. In future, improvement on LZW technique has to be done for better performance of storing string table.

References :

1. Suman M. Choudhary, Anjali S. Patel, Sonal J. Parmar, *"Study of LZ77 and LZ78 Data Compression Techniques"*, International Journal of Engineering Science and Innovative Technology (IJESIT) Volume 4, Issue 3, May 2015

2 Ranjeet S. Pisal "*Implementation of Data Compression Algorithm for Wireless Sensor Network using K-RLE",* International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE) Volume 3, Issue 11, November 2014

3. Dr. Shabana Mehfuz , Usha Tiwari, *"Recent Strategies of Data compression in Wireless Sensor Networks",* Proc. of Int. Conf. on Advances in Electrical & Electronics, AETAEE, Elsevier, 2013

4. Shahina Sheikh, Ms. Hemlata Dakhore, *"Data Compression Techniques for Wireless Sensor Network",* (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 6 (1) , 2015, 818-821

5. Amit Jain, Kamaljit I. Lakhtaria Sir Padampat, *"Comparative Study Of Dictionary Based Compression Algorithms On Text Data"*, International Journal of Computer Engineering and Applications, Volume VI, Issue II, May 14 www.ijcea.com ISSN 23213469.

6. A Jeevarathinam, K Lakshmi, K Thilagam, K Rama, *"Overview Of Tenet: Architecture For Tiered Sensor Networks"* International Journal of Engineering Science and Technology 1 (3), 379-387 Vol. 3 No. 1 Jan 2011.

7. A.Jeevarathinam, Dr.S.Manju Priya· *"Comparison On Hierarchical Routing Protocols In WSN"* Karpagam JCS Vol.10 Issue 5 July - August 2016..

8. Chetna Bharat Mudgule, Prof. Uma Nagaraj, Prof. Pramod D. Ganjewar , *"Data Compression in Wireless Sensor Network: A Survey"*, International Journal of Innovative Research in Computer and Communication Engineering Vol. 2, Issue 11, November 2014

9. Upasana Mahajan, Dr. Prashanth C.S.R,. *"Algorithms for Data Compression in Wireless Computing Systems"*, IJCSI International Journal of Computer Science Issues, Vol. 10, Issue 5, No 1, September 2013

10. Vandana Jindal, A. K. Verma & Seema Bawa, *"Impact of Compression Algorithms on Data Transmission"* International Journal on Advanced Computer Theory and Engineering (IJACTE), ISSN (Print) : 2319 – 2526, Volume-2, Issue-2, 2013

11. Amit Jain, Kamaljit I. Lakhtaria, *"Comparative Study Of Dictionary Based Compression Algorithms On Text Data"*, International Journal of Computer Engineering and Applications, Volume VI, Issue II, May 14