International Journal of Nanoelectronics and Materials
Volume 14 (Special Issue) December 2021 [191-197]

Run-Length Encoding (RLE) Data Compression Algorithm Performance Analysis on Climate Datasets for Internet of Things (IoT) Application

Nor Asilah Khairi[1] and Asral Bahari Jambek[1,*]

[1]Faculty of Electronic Engineering Technology, Universiti Malaysia Perlis, Kampus Alam UniMAP, Pauh Putra, 02600 Arau, Perlis, Malaysia

ABSTRACT

Wireless sensor nodes play an important role for Internet of Things (IoT) applications. However, these devices often come with limited memory sizes and battery life. Thus, to overcome these problems, this work focuses on studying the data compression algorithm suitable for wireless sensor nodes. In this work, run-length encoding (RLE) compression algorithm performance is studied, especially when compressing various climate datasets. This dataset includes temperature, sea-level pressure, air pollution index, and water level. In our experiment, the RLE algorithm gives the best compression ratio for temperature and sea-level pressure, with 0.62 and 0.63 compression ratios, respectively. These are equivalent to around 40% data saving. For air pollution index and water level dataset, our experiment gives 0.96 and 0.93 compression ratios, respectively. Since this data has a low number of repetitive values, the RLE achieves around 10% saving for this kind of data.

Keywords: Data Compression, Run Length Encoding, Internet of Things

1. INTRODUCTION

Internet of Things (IoT) allows various objects and people to be connected through the wireless network [1, 2]. In 2008, the internet connected more things than people [3], while in 2020, 50 billion connections are part of the IoT [4]. Researchers believed that while these devices generate low-rate traffic, it has become a significant challenge nowadays for network providers and the internet as a whole [5, 6]. Furthermore, these billions of connected devices are producing a massive amount of data to be stored.

Furthermore, sensor nodes have limited energy supply and low memory capacity due to their small batteries and memory size. Thus, the sensor nodes must operate as low energy as possible [7, 8]. The transmission module has been identified as the most significant energy consumer in the sensor node. This is because it requires a large amount of energy to power up the wireless transmitter [9]. To overcome these problems, embedding a data compression algorithm into IoT sensor nodes is highly recommended [7]. In this paper, the run-length encoding (RLE) data compression algorithm is studied, and its performance is measured against various climate datasets.

This paper is organized as follows. Section 2 discusses the existing RLE data compression algorithm and its variants. Section 3 discusses the experimental work done in this paper. The results of the RLE performance are discussed in Section 4. Finally, Section 5 concludes this paper.

*asral@unimap.edu.my

2. LITERATURE REVIEW

RLE algorithm compresses data by removing the data redundancy in a dataset [10]. The algorithm works as follows. If a data item d occurs n consecutive times in the input stream, it replaces the n occurrences with a pair symbol nd. For example, RLE converts XXXXXXX characters into X7.

An improved RLE was used in [10], called Aggregated Deflate RLE (ADR). The authors embedded the algorithm into the wireless sensor body network. The paper reports that the proposed method can obtain a better compression ratio compared to the conventional RLE. A combination of RLE and ASCII (RL_ASCII) methods was used in [11] to compress DNA data. In this method, a modified version of RLE is performed first, followed by performing a data encoder on the binary bits. This algorithm can produce a good compression ratio on DNA data, especially when compressing the genome of mitochondria data. This comes at the expense of higher execution time.

The author in [12] focused on increasing the efficiency of data transfers between the main memory and co-processor. In this work, the researchers used the Intel Xeon Phi and NVidia GPU as the hardware platform and combined the RLE and Null Suppression as the compression algorithm. Based on the results, these methods reduce the amount of time transferring data from the main memory to the device.

3. METHODOLOGY

This section details the RLE implementation in this work. The algorithm is written and compiled using C programming. The program reads the dataset character by character and uses a unique symbol when it found repeating data. For example, the program compresses XXXXXXX to X@7, where @ is a symbol that pre-fix the number of repetitions. In this example, if each character is represented by 8-bit (1 byte), the implemented algorithm compresses the original data from 7-byte to 3-byte. In this work, the program only compresses characters with more than 2 repetition values to prevent an unnecessary increase in compressed file size.

Figure 1 shows the flow chart of the RLE data compression implementation used in this work. The data is read from a file. First, a character is read and stored into a first array called array1. Next, the data inside array1 is compared with the following input character. If the character is not the same, it will be stored in a second array, called array2. However, if the character is the same, it will count the number of the data repeated and store the duplicate data in array2. If the repetitive count is more than 2, it will store the data together with the repeat symbol and the repeat count in the array2. This process is continued for every character in the files until it reaches the end of the file. Next, the compressed data is output into a text file.

Figure 2 shows the process of decompressing the data using RLE. First, the program reads a text file that consists of the compressed data. The read characters data is put into an array. Then the character in the array will be scanned. If there is no repetitive symbol, the character will be stored in array2. If there is a repetitive symbol, the character will be duplicated based on the number of repetitions and stored in the array2. This process continues until it reaches the end of the file. Finally, the RLE program will output the decompressed data into a text file format.

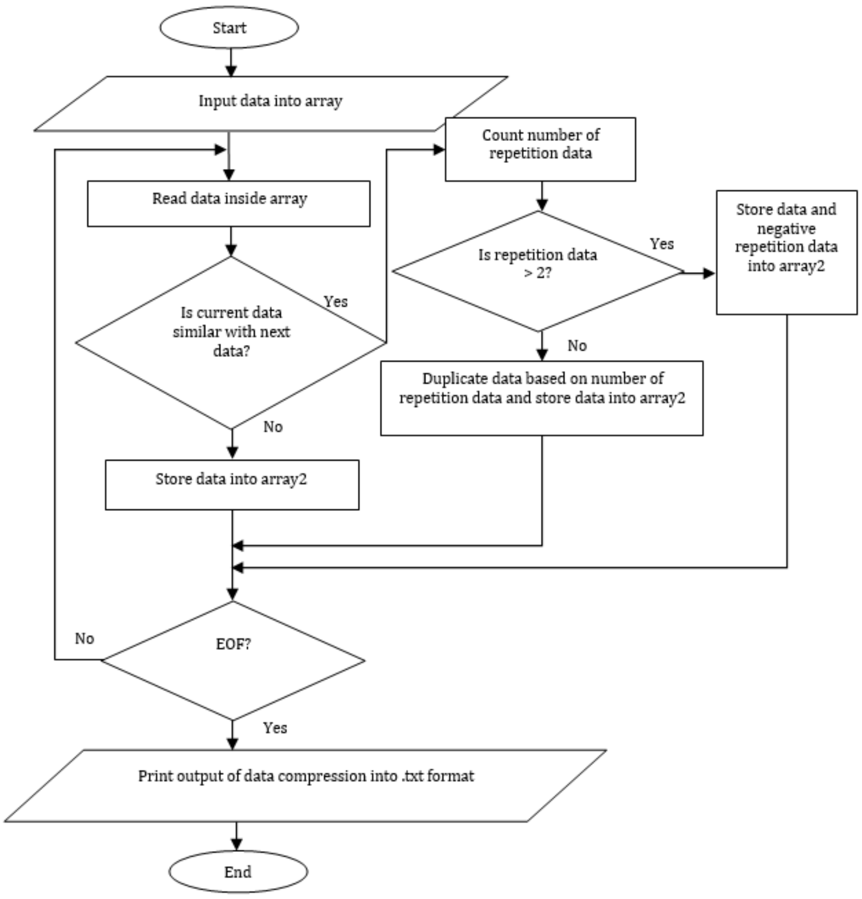

Start
Input data into array
Read data inside array
Is current data similar with next data?
Yes
No
Count number of repetition data
Is repetition data > 2?
Yes
No
Store data and negative repetition data into array2
Duplicate data based on number of repetition data and store data into array2
Store data into array2
EOF?
No
Yes
Print output of data compression into .txt format
End

Figure 1. Flowchart of RLE compression algorithm

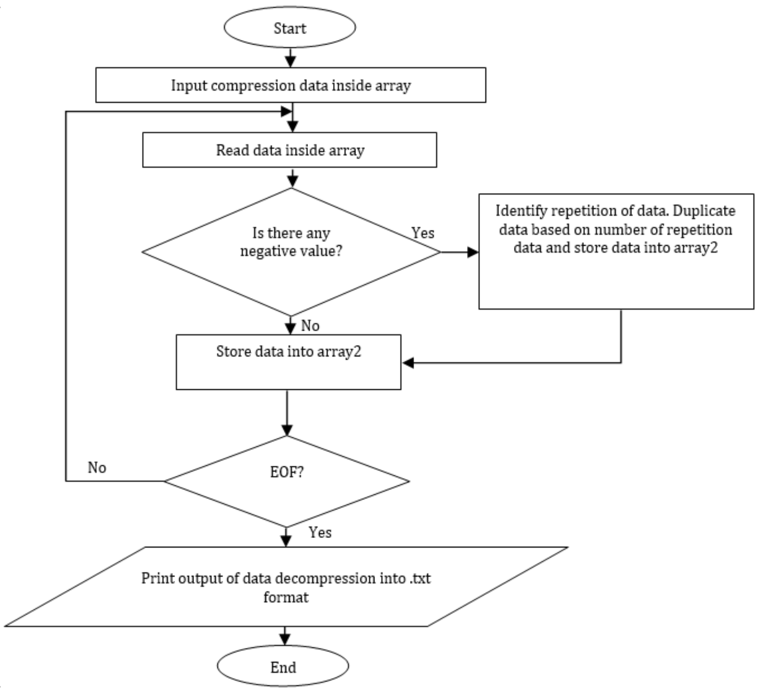
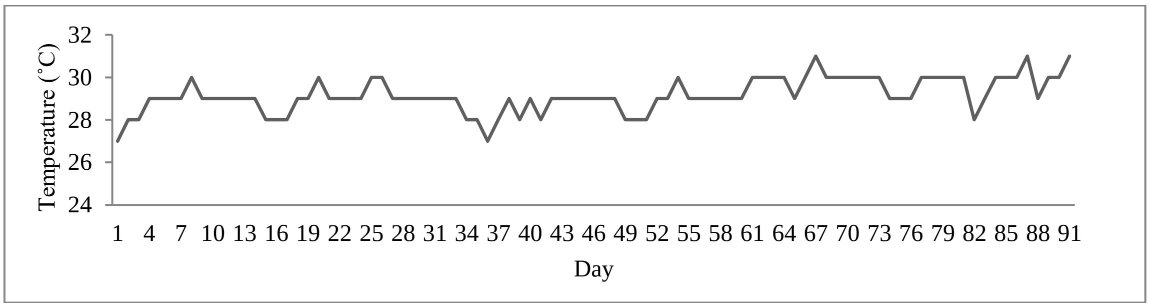
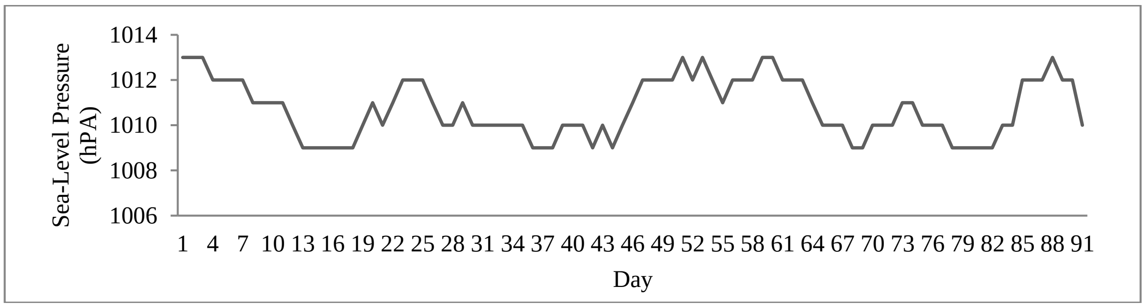

Start
Input compression data inside array
Read data inside array
Is there any negative value?
Yes
No
Identify repetition of data. Duplicate data based on number of repetition data and store data into array2
Store data into array2
EOF?
No
Yes
Print output of data decompression into .txt format
End

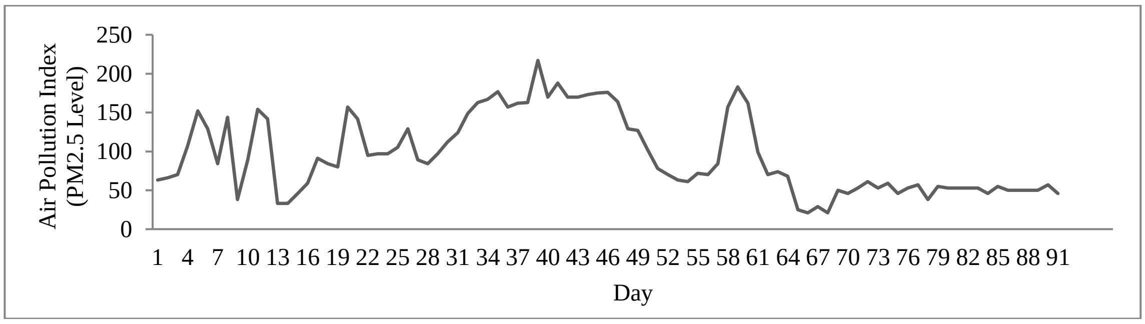
Figure 2. Flowchart of RLE decompression algorithm



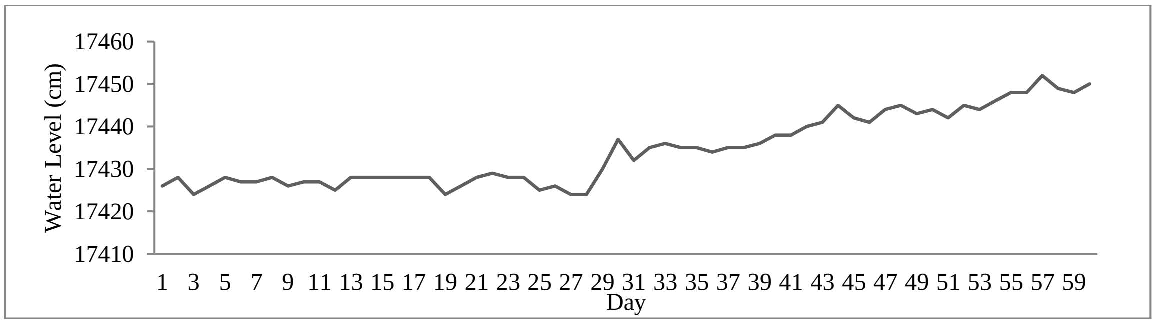
(a)



(b)



(c)



(d)

Figure 3. Graph of datasets for (a) Average temperature measurements at the Kuala Lumpur International Airport (TEMP) (b) Sea-level pressure measurement at the Kuala Lumpur International Airport (SLP) (c) China PM2.5 Report for air pollution index in Beijing (API) (d) Dataset from US Army Corps of Engineers dataset for measurement of the water level in Lake Erie (WL)

To measure the RLE compression performance, climate datasets from various application domains are selected in this study. Two datasets from Weather Underground [13] are selected, such as average temperature measurements at the Kuala Lumpur International Airport (TEMP) and sea-level pressure measurement at the Kuala Lumpur International Airport (SLP). The dataset from China PM2.5 Report for air pollution index in Beijing (API) [14] and the dataset from the US Army Corps of Engineers dataset for measurement of the water level in Lake Erie (WL) [15] are also used in this study. These datasets are selected because they represent different data patterns and a variety of repetitive data. Figure 3 illustrates the graphs of these four datasets.

4. RESULT AND DISCUSSION

This section discusses the RLE algorithm's compression results on various datasets: TEMP, SLP, API, and WL. The compression ratio is an important metric in estimating the performance of the compression algorithm. The compression ratio is calculated based on equations (1) [16]. A lower compression ratio means better compression performance.

$$\text{Compression ratio} = \frac{\text{Compress size}}{\text{Original size}} \tag{1}$$

Figure 4 shows the file size before and after compression, while Figure 5 highlights the equivalent compression ratio for each dataset. Based on the results, all datasets' file sizes decrease after the compression. Based on the figures, the RLE performs well for TEMP and SLP datasets, where it achieves 0.62 and 0.63 compression ratios for TEMP and SLP datasets, respectively. This is equivalent to a reduction of almost 40% of the original file size. On the other hand, RLE can only achieve 0.96 and 0.93 compression ratio for the API and WL datasets, respectively. This is equivalent to less than 10% data saving from the original data size. Based on the results, the RLE compression algorithm is suitable for data that has high repetition, such as shown in TEMP and SLP. RLE is less effective for API and WL due to the low data repetition in this dataset.

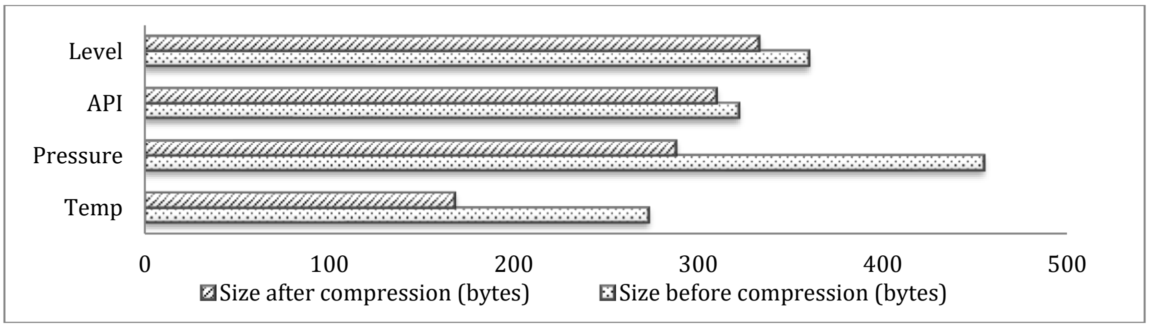


Figure 4. Size before and after compression for four datasets

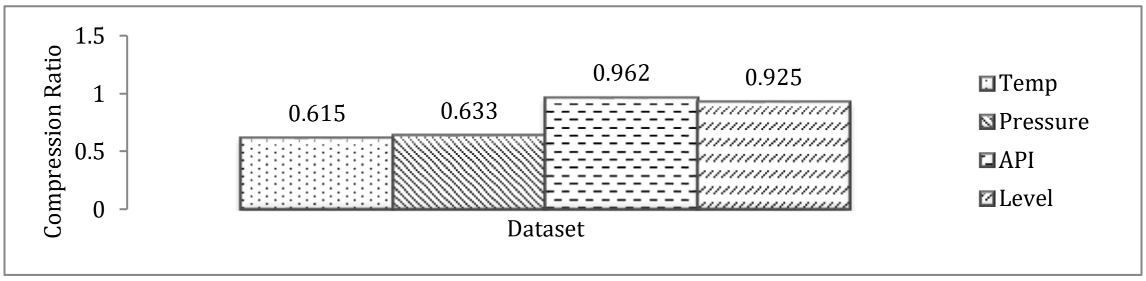


Figure 5. Compression ratio on four datasets

5. CONCLUSION

Wireless sensor nodes are an important device in the IoT. However, these devices are typically operating with small battery sizes and memory. Thus, these devices must consume minimal energy consumption and data storage. To solve this problem, RLE data compression performance is studied in this work. Several types of datasets are analyzed to measure the performance of the algorithm. This dataset is compressed, and its compression ratio is analyzed. Based on the experimental results, RLE provided a 0.62 and 0.63 compression ratio for the TEMP and SLP datasets, respectively. This is equivalent to almost 40% data saving. Furthermore, this saving is due to high data repetition in these datasets. On the other hand, for API and WL datasets, the algorithm can only achieve 0.96 and 0.93 compression ratios, respectively. This is equivalent to around 10% data saving due to low repetitive values in the datasets. In the future, this work will focus on increasing the performance of this RLE method by applying double compression to produce a higher compression ratio.

REFERENCES

[1] A. W. Burange, H. D. Misalkar, "Review of Internet of Things in development of smart cities with data management & privacy," in 2015 International Conference on Advances in Computer Engineering and Applications (ICACEA), (2015) pp. 189 - 195.

[2] S. Abraham, J. Beard, R. Manijacob, "Remote Environmental Monitoring Using Internet of Things (IoT)." 2017 IEEE Global Humanitarian Technology Conference (GHTC), (2017).

[3] S. E. Collier, "The Emerging Enernet: Convergence of the Smart Grid with the Internet of Things," in 2015 IEEE Rural Electric Power Conference (REPC), (2015) pp. 65-68.

[4] M. A. Burhanuddin, A. A. Mohammed, R. Ismail, H. Basiron, Internet of Things Architecture: Current Challenges and Future Direction of Research: International Journal of Applied Engineering Research, vol **12** issue 21, (2017) pp.11055-11061.

[5] S. Vural, P. Navaratnam, N. Wang, C. Wang, L. Dong, R. Tafazolli, "In-network caching of Internet-of-Things data," in 2014 IEEE International Conference on Communications (ICC), (2014) pp. 3185 - 3190.

[6] N. Al-Falahy, O.Y.K Alani, Supporting Massive M2M Traffic in the Internet of Things Using Millimetre Wave 5G Network. 2017 9th Computer Science and Electronic Engineering (CEEC), (2017).

[7] S. Sheikh, H. Dakhore, Data Compression Techniques for Wireless Sensor Network: (IJCSIT) International Journal of Computer Science and Information Technologies, vol **6** issue 1 (2015) pp. 818-821.

[8] N. Saleh, A. Kassem, A. M. Haidar Energy-Efficient Architecture for Wireless Sensor Networks in Healthcare Applications: IEEE Access, (2018).

[9] A. B. Jambek, N. A. Khairi, Performance Comparison of Huffman And Lempel-Ziv Welch Data Compression for Wireless Sensor Node Application: American Journal of Applied Sciences, vol **11** issue 1, (2014) pp. 119-126.

[10] B. Tiwari, A. Kumar, "Aggregated Deflate-RLE compression technique for body sensor network," in 2012 CSI Sixth International Conference on Software Engineering (CONSEG), (2012) pp. 1 - 6.

[11] Priyanka, S. Goel, "A compression algorithm for DNA that uses ASCII values," in 2014 IEEE International Advance Computing Conference (IACC), (2014) pp. 739 - 743.

[12] K. Y. Besedin, P. S. Kostenetskiy, S. O. Prikazchikov, "Using Data Compression for Increasing Efficiency of Data Transfer Between Main Memory and Intel Xeon Phi Coprocessor or NVidia GPU in Parallel DBMS," in 4th International Young Scientist Conference on Computational Science, (2015) pp. 635–641.

[13] Weather Underground, "https://www.wunderground.com".

[14] China PM2.5 Report, "http://young-0.com/airquality/".

[15] US Army Corps of Engineers, "http://www.lre.usace.army.mil/Home.aspx".

[16] D. Salomon, “Data Compression The Complete Reference 3rd ed.” New York, NY: Springer-Verlag New York, Inc, (2004) pp.10.