International Journal of Electrical and Computer Engineering (IJECE)
Vol. 15, No. 2, April 2025, pp. 2468~2478
ISSN: 2088-8708, DOI: 10.11591/ijece.v15i2.pp2468-2478

D-RAKE compression for enhanced internet of things data management in air quality monitoring

Kartika Sari, Rahmi Hidayati
Department of Computer System Engineering, Faculty of Mathematics and Natural Sciences, Universitas Tanjungpura, Pontianak, Indonesia

Article Info

Article history:

Received Jun 25, 2024
Revised Sep 18, 2024
Accepted Oct 1, 2024

Keywords:

Air quality monitoring
Data compression
D-RAKE
Internet of thing
Lossless compression

ABSTRACT

This study addresses the issue of air pollution in Pontianak, marked by high levels of pollutant particles and chemical compounds that cause respiratory health risks. The research involves essential air quality monitoring using various sensors for temperature, humidity (DHT22), O_2 (MQ-135), CO (MQ-7), CO_2 (MG-811), and dust (GP2Y1010AU0F), collected real-time, leading to a notable increase in data volume. Due to limitations in internet of things (IoT) devices, there is a need for integration between cloud and IoT through data transmission to reduce the communication time and memory usage. The escalation in sensor data volume requires a lossless compression technique to ensure efficient storage without sacrificing crucial information. Compression plays a vital role in overcoming complex storage challenges, facilitating real-time data access for monitoring, and contributing to sustainable efforts to improve air quality in Pontianak. This research applies the D-RAKE compression method based on basic counting procedures with minimal memory requirements, cost-effective, low-speed microcontrollers commonly used in IoT devices. Despite its simplicity, simulation results indicate that the D-RAKE algorithm outperforms well-established compression methods such as *gzip*, *bzip*2, and *rar*, particularly for data sequences with sparse elements. Moreover, when applied to real-world data, D-RAKE achieves superior compression ratios compared to IoT-focused compression techniques.

Corresponding Author:

Kartika Sari
Department of Computer System, Faculty of Mathematics and Natural Sciences, Tanjungpura University
Jl. Prof. Dr. H. Hadari Nawawi, Bansir Laut, Pontianak Tenggara, Pontianak, West Kalimantan, Indonesia
Email: kartika.sari@siskom.untan.ac.id

1. INTRODUCTION

Pontianak, like many other major cities, faces significant challenges due to air pollution. The presence of dust particles and harmful chemicals such as carbon monoxide poses serious risks to respiratory health, making air quality monitoring an essential task [1]–[10]. The large volumes of data generated by air quality monitoring systems require advanced data compression techniques to manage the data efficiently [11]–[19]. This challenge is further compounded by the limitations of internet of things (IoT) devices, which rely on cloud transmission to reduce communication time and memory usage [20]. While timely sensor data is crucial for effective monitoring, the massive data volumes can lead to throttling issues, delaying the transmission and processing of data. Batch processing, where data is collected periodically before transmission, provides a partial solution. However, the continuous generation of data increases batch size, reducing the efficiency of data delivery, processing, and storage. Thus, sophisticated data compression methods that preserve critical information are essential for effectively managing the growing data volume from Pontianak's air quality monitoring systems [21]–[26]. Data compression addresses storage challenges, enables real-time data access for analysis, and supports efforts to improve air quality in Pontianak. Integrating compression into monitoring systems enhances storage efficiency and ensures that critical data remains available for analysis and informed decision-making [27], [28].

Several previous studies have explored various data compression techniques in environmental monitoring, each contributing uniquely to the advancement of this field [29]–[33]. For instance, Gunawan *et al.* [34] designed and implemented a portable outdoor air quality measurement system using Arduino but without applying data compression, which led to challenges in data management and cloud transmission. Wu *et al.* [35] proposed a method for data compression to alleviate the sensitivity packet losses in wireless sensor network (WSN). In addition, Hossain and Roy [36] did research on data compression for IoT sensors to optimize storage using lossless data compression and achieved a compression efficiency of 50%; but it encountered some errors with the original data changing by 0% to 1.5% after decompression. Minewaki *et al.* [37] conducted research on lossless compression algorithms for environmental data using ZS.Q (zero-skip quantization); but, these methods do not fully satisfy the near-lossless (NL) condition. Ramalingam *et al.* [38] delved into the benefits of data compression in real-time data transmissions and fault analysis. Hwang *et al.* [39] proposed a bit depth compression (BDC) technique to compress the sensor data. Some studies highlight that while data compression is crucial for environmental monitoring, achieving optimal compression efficiency without compromising data integrity is still challenging. The difficulties in optimizing these techniques for IoT sensor data under various conditions indicate a need for further research.

This study addresses these challenges by introducing a modified D-RAKE method to improve compression efficiency and ensure 100% accuracy in data decompression. Unlike previous methods, the D-RAKE approach further develops the ability to effectively manage large volumes of sensor data in air quality monitoring, ensuring that critical information is preserved while significantly reducing data size. This novel contribution provides a solution that addresses the existing limitations of current compression techniques, particularly in the context of large-scale air quality monitoring. The subsequent sections of this manuscript will detail the proposed D-RAKE method, including the specific modifications made to the compression algorithm. Simulation and experimental results will be presented to demonstrate the method's superiority over other established lossless compression techniques, such as *gzip*, *bzip*2, and *rar*. The discussion will focus on the context of air quality monitoring in Pontianak for supporting pollution reduction initiatives in other major cities. By offering a more efficient and effective data compression solution, this study aims to enhance the overall effectiveness of air quality monitoring efforts, contributing to improved public health and environmental sustainability.

2. THE PROPOSED METHOD

2.1. D-RAKE method

The D-RAKE data compression algorithm is a development of RAKE's method [40] that is used to compress data while retaining all the information, and it is particularly efficient when applied to binary sequences that have a low density of data points. In cases where there are minimal variations in time for signals, the algorithm can achieve compression by processing consecutive differences between samples, which are represented as the residue $r(i) = x(i) - x(i-1)$. The D-RAKE algorithm can be utilized when the data to be compressed contains fewer than 15% of 1s to the total number of bits. Nevertheless, if the data consists of more than 40% of 1 s, the compression method cannot be effectively applied. The D-RAKE algorithm operates according to the rules: i) A codeword of '0' means all bits in the RAKE are '0', indicating no '1' bits are present; ii) A codeword of $L = 1 + \lceil log_2 T \rceil$-bits means at least one '1' bit is found in the RAKE. The first bit of the codeword is set to '1' to indicate the presence of a '1' bit, and the remaining $\lceil \log_2 T \rceil$ bits encode its position, $PfirstP\_\{first\}P_{first}$, which ranges from 0 to T−1; iii) The D-RAKE shifts by P_{first} +1 if a '1' bit is found, or by T if no '1' bit's found; iv) This process repeats until all bits to be compressed are processed; and v) The final compressed sequence is obtained by concatenating all the codewords.

2.2. D-RAKE for data compression and decompression

The D-RAKE method is designed to optimize data compression that requires efficient data management like in IoT environments. This method reduces the large datasets size while preserving their integrity, making it ideal for handling the substantial sensor data typically generated in real-time monitoring systems. Figure 1(a) and Figure 1(b) shows the D-RAKE data compression and decompression process.

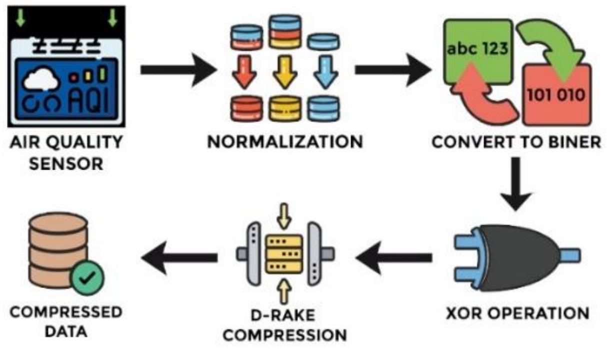
Figure 1(a) and Figure 1(b) represent the simulation of D-RAKE algorithm. The explanation of Figure 1(a) data compression and Figure 1(b) data decompression process is as:

– Normalization: combine all sensor values and the timestamp into a single decimal value.

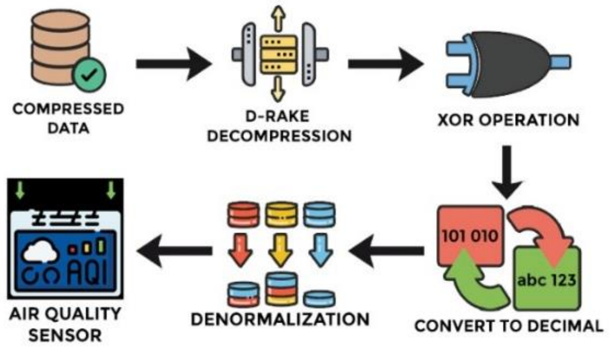
– Convert to binary: convert the normalized decimal value to binary.
– XOR operation: perform XOR operation between the binary data and default bits stored on the server.
– D-RAKE compression: compress the XOR result using the RAKE compression algorithm.

After compressing the data, it will transmit it to the cloud. Following this, the cloud will undertake the data decompression process to guarantee the complete restoration of data for use or display. The data decompression procedure involves normalizing the data and reverting it to its original format. The data decompression process is delineated as:

– D-RAKE decompression: decompress the data using the RAKE decompression algorithm.
– XOR operation: perform XOR operation between the decompressed data and the default bits to get the original binary data.
– Convert to decimal: convert the original binary data back to decimal.
– Denormalization: extract the original sensor data and timestamp from the denormalized value.



(a)



(b)

Figure 1. The simulation of D-RAKE algorithm: (a) data compression process and (b) data decompression process

3. METHOD

The data in this study serves as input for the developed system, which is then processed and utilized to generate output. Specifically, the collected data includes air quality values obtained from the air quality monitoring system, which are subjected to compression. Additionally, the study records the size of the sensor data values both before and after compression to evaluate the effectiveness of the compression process.

3.1. Data collection

The data collection phase involves gathering air quality parameters from multiple sensors: DHT22 for temperature and humidity, MQ-135 for oxygen (O_2), MQ-7 for carbon monoxide (CO), MG-811 for carbon dioxide (CO_2), and GP2Y1010AU0F for dust particles. These measurements, along with a timestamp, form the basis of the data for subsequent compression and analysis. Algorithm 1 presents the steps involved in this process.

Algorithm 1. Applying the data collection
```
function collect_sensor_data():
timestamp=get_current_timestamp()
temperature=read_sensor(DHT22)
humidity=read_sensor(DHT22)
O2=read_sensor(MQ-135)
CO=read_sensor(MQ-7)
CO2=read_sensor(MG-811)
dust=read_sensor(GP2Y1010AU0F)
sensor_data=[temperature, humidity, O2, CO, CO2, dust]
return timestamp, sensor_data
```

Algorithm 1 outlines the steps as follows: i) The system reads sensor data for each parameter and stores them in an array called $sensor\_data$; ii) A timestamp is added to each collection cycle for tracking purposes; iii) Sensor data for temperature, humidity, O_2, CO, CO_2, and dust particles is read using the $read\_sensor(sensor\_name)$ function and stored in an array called $sensor\_data$; and iv) The function returns both the timestamp and the collected sensor data for subsequent processing.

3.2. Applying the proposed method to air quality monitoring system
3.2.1. Data compression algorithm based on D-RAKE

Data compression in this system is performed using a data encoding technique based on ASCII for converting sensor data, which is in character form, into binary form, and a data modeling technique using XOR operations on the sensor's binary data. Algorithm 2 applying the proposed method to compress the air quality monitoring system's data:

Algorithm 2. Applying the proposed method to compress the air quality monitoring system's data
```
function rake_compress(timestamp, sensor_data):
default_data=None
# Step 1: Normalize and encode sensor data
   binary_data=""
   for value in sensor_data:
      binary_value=ascii_to_binary(value)
      binary_data += binary_value
# Step 2: Check if this is the first data
   if is_first_data():
      default_data=binary_data
      store_default_data(default_data)
   else:
      default_data=get_default_data()
      binary_data=xor_operation(binary_data, default_data)
# Step 3: RAKE Compression
   compressed_data=rake_algorithm(binary_data)
return compressed_data
```

The D-RAKE-based compression algorithm reduces the size of the collected data while preserving its accuracy. The process involves:

a. Normalization and encoding: sensor values are converted into binary form using ASCII encoding.
b. XOR operation: if the current data is not the first data point, the system applies an XOR operation with a default (previously stored) binary dataset to identify changes, reducing redundancy.
c. RAKE compression: the processed binary data undergoes the RAKE compression algorithm, which minimizes the data size.

The compressed data is returned and ready for transmission to the cloud. This approach ensures that only significant changes in the data are stored, optimizing memory usage and transmission bandwidth.

3.2.2. Data transmission

After the sensor data values are successfully compressed, the next step involves transmitting this compressed data to a cloud storage system for further processing. Once the data is stored in the cloud, decompression can be performed to restore the data to its original form, making it ready for use or display. Algorithm 3 is for the data transmission:

Algorithm 3. The data transmission
```
function transmit_data_to_cloud(compressed_data):
cloud_store(compressed_data)
```

The compressed data is transmitted to a cloud storage system for analysis and visualization. The transmission process is straightforward:

a. The $transmit\_data\_to\_cloud()$ function handles the process of uploading compressed data to the cloud storage.
b. The $transmit\_data\_to\_cloud()$ function uses the $cloud\_store()$ function to upload compressed data to the cloud securely, ensuring centralized storage and accessibility for further processing.

3.2.3. Data decompression algorithm based on D-RAKE

The data decompression process is crucial in ensuring that compressed data can be accurately restored to its original form without any loss of information or precision. This step is particularly important in applications such as air quality monitoring, where accurate data is essential for analysis and decision-making. Algorithm 4 implements the proposed method for decompressing air quality monitoring data, which involves three key stages: RAKE decompression to reverse the compression process, normalization to reconstruct the original binary values, and binary-to-ASCII conversion to translate the binary data back into usable sensor readings. These steps ensure that the decompressed data is accurate for further use.

Algorithm 4. Applying the proposed method to decompress the air quality monitoring system's data
```
function rake_decompress():
compressed_data=retrieve_from_cloud()
# Step 1: RAKE Decompression
   binary_data=rake_decompression_algorithm(compressed_data)
# Step 2: Normalization using XOR with default data
   default_data=get_default_data()
   original_binary_data=xor_operation(binary_data, default_data)
# Step 3: Convert binary to original sensor data
   sensor_data=binary_to_ascii(original_binary_data)
return sensor_data
function rake_decompression_algorithm(compressed_data):
binary_data=compressed_data # Placeholder for the algorithm
return binary_data
function binary_to_ascii(binary_data):
ascii_data=""
for i in range(0, len(binary_data), 8):
byte=binary_data[i:i+8]
ascii_data += chr(int(byte, 2))
return ascii_data.split()
function ascii_to_binary(value):
binary_value=""
for char in str(value):
binary_value += format(ord(char), '08b')
return binary_value
function xor_operation (data1, data2):
return ''.join (['1' if b1 != b2 else '0' for b1, b2 in zip(data1, data2)])
function rake_algorithm(data): # Implement RAKE compression algorithm
compressed_data=data      # for the actual RAKE
return compressed_data
```

Algorithm 4 explains that the data decompression process consists of three main stages:
a. Retrieving compressed data: The process begins by retrieving compressed data from cloud storage using the $retrieve\_from\_cloud()$ function. Once the data is successfully retrieved, the first stage, RAKE decompression, is performed. The $rake\_decompression\_algorithm()$ function is used to reverse the RAKE compression logic, converting the compressed binary data back into its decompressed binary form.
b. Data normalization: In the second stage, normalization is performed by reconstructing the original binary values using an XOR operation between the decompressed binary data and a default dataset. This step is executed using the $xor\_operation()$ function, which restores the data to its original state prior to compression. The default dataset serves as a reference to accurately reverse any transformations applied during the compression process.
c. Binary-to-ASCII conversion: The third and final stage involves converting the reconstructed binary data back into the original sensor values using binary-to-ASCII conversion. The $binary\_to\_ascii()$ function processes the binary data in 8-bit chunks (bytes) and converts them into their corresponding ASCII characters. The resulting data is then split into individual sensor values, completing the decompression process.

Through these steps, Algorithm 4 ensures that compressed data can be accurately restored without any loss of information. This is crucial for applications that demand high precision, such as air quality monitoring systems.

3.3. Data testing

The system utilizes a structure consisting of input, processing, and output. Testing begins by inserting sensor data into the system to evaluate its performance. During the processing phase, data is compressed and decompressed. The test results are obtained by examining the data that has been compressed and transmitted through the IoT gateway to the cloud. Each parameter, including data size, compression, decompression efficiency, and the compression and decompression ratio, is then evaluated to assess the test results. Figure 2 provides a visual representation of the test block diagram.

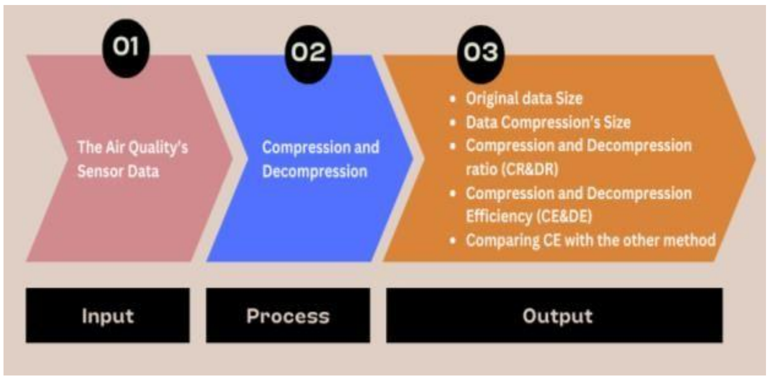

01
The Air Quality's Sensor Data
02
Compression and Decompression
03
Original data Size
Data Compression's Size
Compression and Decompression ratio (CR&DR)
Compression and Decompression Efficiency (CE&DE)
Comparing CE with the other method
Input
Process
Output

Figure 2. The test block diagram

3.4. Evaluating the performance of text file data compression algorithms

Once the testing system has been set up, the subsequent step involves conducting tests and measurements on the implemented system. Afterward, a comprehensive analysis is carried out to determine if the system aligns with the initial plan. Evaluating the performance of text file data compression algorithms are about compression-decompression ratio and compression-decompression efficiency [41]–[43].

a. Compression ratio

Compression ratio (CR) is a measure that quantifies the relationship between the number of bits before compression and after compression. The formula for calculating the CR is presented in (1).

$$CR = \frac{Number\ of\ Bits\ BEFORE\ Compression}{Number\ of\ Bits\ AFTER\ Compression} \tag{1}$$

b. Compression efficiency

Compression efficiency (CE%) refers to the effectiveness of a compression algorithm or technique in reducing the size or volume of data while retaining its essential information or quality. It is a measure of how well the compression process reduces the number of bits or bytes needed to represent the data [44].

$$CE\% = 100 \times (1 - \frac{1}{CR}) \tag{2}$$

The CE is presented in percentages to describe a measure of data compression's success.

3.5. Evaluating the performance of text file data compression algorithms

Before delving into the technical metrics, it is essential to establish the importance of evaluating compression and decompression processes. These metrics provide insights into the effectiveness of an algorithm in reducing data size while preserving its integrity. Two critical measures used for this evaluation are the decompression ratio (DR) and decompression efficiency (DE).

a. Decompression ratio

DR is determined by comparing the number of bits before and after decompression. The formula for calculating the decompression ratio is illustrated as (3).

$$DR = \frac{\text{Number of Bits BEFORE Compression}}{\text{Number of Bits AFTER decompression}} \tag{3}$$

b. Decompression efficiency (DE%) is shown in (4).

The decompression efficiency (DE%) further evaluates the algorithm's performance by expressing the effectiveness of decompression as a percentage. It is determined using (4):

$$DE\% = 100\ x\ (1 - \frac{1}{DR}) \quad (4)$$

DE is expressed as a percentage, representing a metric that quantifies the effectiveness of data decompression.

4. RESULTS AND DISCUSSION

This study reveals that the D-RAKE algorithm greatly improves data compression efficiency in air quality monitoring over traditional methods. Experimental results show that D-RAKE can reduce data size by up to 68.67% while maintaining the integrity of essential information. This higher CE means that more data can be stored and transmitted more quickly, which is crucial for real-time applications in air quality monitoring. This success is supported by direct comparisons of compression efficiency between D-RAKE and other traditional methods.

To further validate these findings, the performance of the D-RAKE algorithm was compared with five other compression methods, namely *rar*, *bzip*2, *gzip*, RAKE, and D-RAKE, using two air quality datasets: Ds1 (Dataset 1), representing indoor air quality data, and Ds2 (Dataset 2), representing outdoor air quality data. This comparison reveals that D-RAKE consistently outperforms the other methods in compression efficiency, particularly with Dataset 1, which has more stable data variations. Traditional compression methods like *rar*, *bzip*2, and *gzip* demonstrated lower performance compared to D-RAKE, especially when handling data with more dynamic variations in Ds2. Meanwhile, RAKE, the predecessor of D-RAKE, also showed good efficiency but still fell short of D-RAKE's performance. These results underscore the superiority of D-RAKE in various scenarios and will be discussed in more detail in the following subsections.

In conclusion, the development of the D-RAKE algorithm represents a significant advancement in data compression for air quality monitoring. The study highlights the algorithm’s ability to improve data storage and transmission efficiency within IoT systems, which can lead to faster response times and more informed decision-making in air quality management. Despite these promising results, several questions remain unanswered, such as how D-RAKE can be further adapted or enhanced to work effectively with various sensor types and different environmental conditions. Future research could focus on refining the D-RAKE algorithm to increase its speed and efficiency, as well as exploring its potential applications in other IoT domains.

4.1. CE parameter

The CE parameter is determined by calculating the CR, taking into consideration the data size before and after compression. In this research, data from sensors were collected across six different time intervals for each testing experiment, yielding diverse data size measurements. This was conducted to assess how the size of the data being compressed affects the CE parameter. Each time interval underwent 50 times collecting data and 6 times testing iterations (T1-T6 times in collecting data sensor), ensuring that the research conclusions could be applied universally to all compressed sensor data. The data compression testing experiments were carried out using a modified version of the D-RAKE data compression algorithm to discern variations in the CE parameter values across. Table 1 shows the outcomes obtained from the sensor data compression tests.

Table 1. CE parameter testing

Trial number (Tn)							
Data size before compression (Bytes)		T1	T2	T3	T4	T5	T6
		1,208	2,416	4,252	7,781	11,645	15,356
Data size after compression (Byte)	MIN	398	767	1,306	2,074	3,396	4,349
	MAX	506	901	1,521	3,176	4,219	6,593
	AVG	488	886,4	1,457.3	2592.8	3,768.5	4,810
CR-compression ratio		2.47	2.73	2.92	3.001	3.09	3.19
CE-compression efficiency (%)		59.6	63.31	65.7	66.67	67.63	68.67

Table 1 presents that the data size before compression gradually increases from T1 to T6, indicating variations in the data sizes used for testing. The data size before compression ranges from 1,208 bytes in T1 to 15,356 bytes in T6. After compression, the data size is significantly reduced, with the smallest compressed data size recorded in T1 at 398 bytes and the largest in T6 at 6,593 bytes. The CR displayed in the table shows how each trial yielded varying ratios, starting from 2.47 in T1 and increasing to 3.19 in T6. This increase in the compression ratio aligns with the increase in the initial data size, indicating that the D-RAKE algorithm becomes more effective when working with larger data sets. The CE also gradually increased from 59.6% in T1 to 68.67% in T6. This suggests that the larger the compressed data, the higher the efficiency achieved by the D-RAKE algorithm. The increase in efficiency demonstrates the D-RAKE algorithm's effectiveness in reducing data size while preserving essential information. To better understand the performance of the D-RAKE algorithm as presented in Table 1, Figure 3 provides a visual representation of CE parameter.

Figure 3 presents the efficiency of sensor data compression employing the D-RAKE compression technique derived from six distinct testing scenarios, covering intervals of 5, 10, 15, 20, 25, and 30 minutes, each repeated 50 times to ensure comprehensive insights applicable across all compressed sensor data. The D-RAKE method demonstrates superior CE compared to various other lossless compression methods. This is primarily attributed to its focus solely on converting sensor data into binary format, disregarding characters, and retaining only the decimal values of default and subsequent sensor data. Consequently, the resulting binary values undergo significant fluctuations with changes in decimal values. For instance, if the humidity sensor records a value of 54.22, this method omits characters, resulting in 5,422. Subsequent conversion yields [1010100101110], followed by normalization to obtain 5,423, then converted to binary form. XOR operation with the default binary value of 5,423, producing [1010100101111], results in numerous '1' bits. Increased discrepancies in decimal data amplify the likelihood of '1' binary occurrences during XOR operations. Conversely, the RAKE algorithm proves more efficient with a higher frequency of binary '0' occurrences.

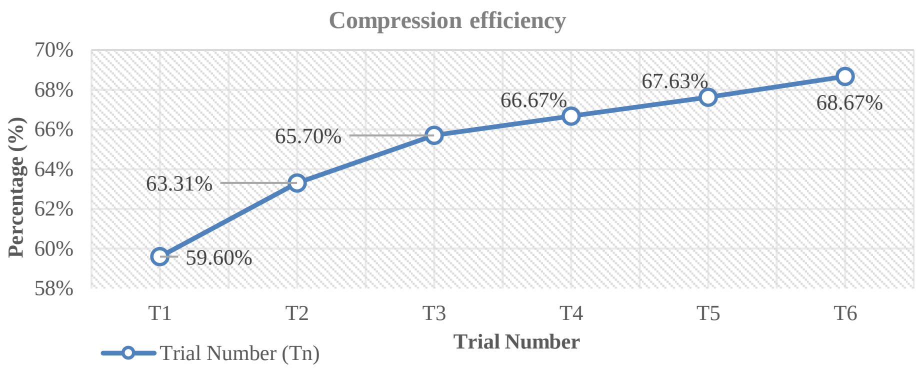


Figure 3. CE parameters

4.2. DE Parameter

During the data decompression testing phase, the D-RAKE method was applied. In each individual test run, data samples were collected over six distinct periods to investigate how the length of the data segment affected the process. After this, the data underwent compression and subsequent decompression to evaluate its ability to return to its original state. Table 2 provides a summary of the data decompression testing using the D-RAKE algorithm.

Based on the data decompression testing outcomes, it was observed that the data length remained constant both before and after decompression, without any alterations. Additionally, when converting the compressed data, the values of the data before and after compression were IDENTICAL and UNCHANGED in all decompression trials. As a result, it can be inferred that the data was entirely and successfully restored to its original state with no loss, achieving a 100% recovery rate.

Table 2. The summary of data compression testing

Trial number (Tn)	Original data size (Byte)	Average data size after compression (Byte)	Data size after decompression (Byte)	DR-decompression ratio (%)
T1	1,208	488	1,208	100
T2	2,416	886.4	2,416	100
T3	4,252	1,457.3	4,252	100
T4	7,781	2,592.8	7,781	100
T5	11,645	3,768.5	11,645	100
T6	15,356	4,810	15,356	100

4.3. Comparison results with some other compression method

The comparison of CE across several data compression methods, including *rar*, *bzip*2, *gzip*, RAKE, and D-RAKE, was conducted using two air quality datasets, Ds1 which represents indoor air quality data, and Ds2 that represents outdoor air quality data. The RAR method shows stable compression efficiency across both datasets, with 28.0% for Ds1 and 36.4% for Ds2. In contrast, *bzip*2 exhibits significant variation, with very low compression efficiency at 4.3% for Ds1, but a substantial improvement to 35.4% for Ds2. The *gzip* method performs similarly to RAR, with identical efficiency values across both datasets. RAKE, a more recent method, demonstrates improved efficiency compared to traditional methods, achieving 32.1% for Ds1 and 40.2% for Ds2. However, D-RAKE, an advanced version of RAKE, stands out as the most effective method, achieving the highest compression efficiencies, with 68.67% for Ds1 and 51.6% for Ds2, making it the most efficient in reducing data size. These findings confirm the effectiveness of the D-RAKE algorithm in efficiently compressing data and reducing data size while preserving essential information. This observation serves as evidence of the algorithm's superiority in improving data compression for air quality monitoring. The comparison between the D-RAKE method and other compression methods, such as *rar*, *bzip*2, *gzip*, and the original RAKE method, is shown in Table 3.

To better understand the performance of different data compression methods, a comparative analysis was conducted using two datasets, Ds1 and Ds2. The CE of various methods, including *rar*, *bzip*2, *gzip*, RAKE, and D-RAKE, was evaluated across these datasets. Figure 3 illustrates the results of this comparison, highlighting the CE (%) achieved by each method in both indoor and outdoor settings. This visual representation provides a clear and concise overview of how each method performs under different environmental conditions, with particular emphasis on the superiority of the D-RAKE algorithm. Figure 4 shows the results.

Figure 4 shows a comprehensive comparison chart, showcasing the CE of several data compression methods tested. This chart is based on two air quality datasets, Ds1 and Ds2. In this chart, it is evident that D-RAKE has the highest CE across both datasets. For Ds1, which represents indoor data, D-RAKE achieves nearly double the CE compared to other methods, demonstrating D-RAKE's ability to compress more stable data with smaller variations effectively. For Ds2, which reflects outdoor data, D-RAKE also shows the best performance, although its CE is slightly lower than for Ds1 due to the greater data variability in outdoor environments. The RAKE method, which is the predecessor of D-RAKE, also shows relatively good performance but still falls short of D-RAKE in terms of CE. Meanwhile, traditional methods such as *rar*, *bzip*2, and *gzip*, though stable, are unable to reach the levels of CE demonstrated by D-RAKE. This figure visually emphasizes the superiority of the D-RAKE method in data compression, especially in the context of air quality monitoring, where reducing data size and preserving essential information are crucial.

Table 3. Compression efficiencies in the case of real-world air quality data

DataSet	CErar (%)	CEbzip2 (%)	CEgzip (%)	CERAKE (%)	CED-RAKE (%)
Ds1	28.0	4.3	28.0	32.1	68.67
Ds2	36.4	35.4	36.4	40.2	51.6

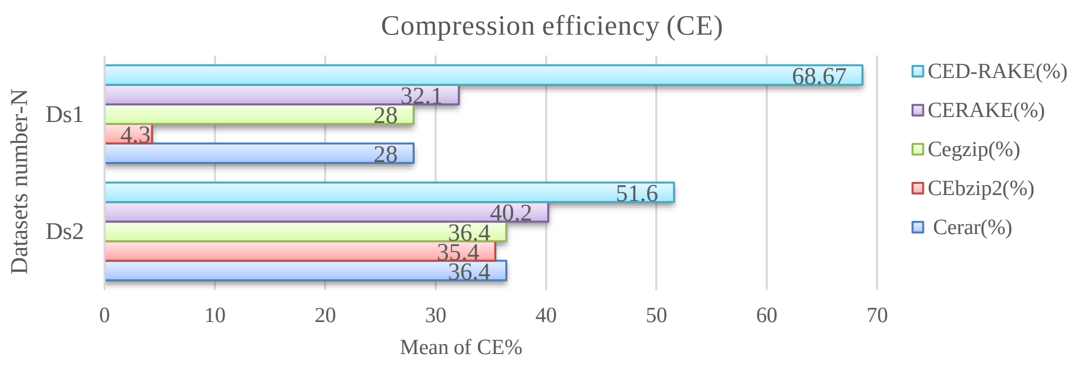


Figure 4. Mean of CE% in the case of real-world air quality data

5. CONCLUSION

This research clearly shows that the D-RAKE algorithm significantly improves data compression efficiency in IoT-based air quality monitoring systems. Our findings reveal that D-RAKE can reduce data size by over 68% while preserving the integrity of critical information. This achievement is particularly important for managing data in IoT environments, where fast data transmission and storage efficiency are crucial for real-time applications. The comprehensive comparison of methods in this study confirms that D-RAKE consistently outperforms other compression methods. Its ability to reduce data size while maintaining important information makes it a standout solution for air quality monitoring. However, there are still some questions to explore, such as how the algorithm can be further optimized to work with different types of sensors and under various environmental conditions. Although D-RAKE is highly efficient, its complexity could affect processing speed, which needs further investigation. Looking ahead, future work will focus on refining the algorithm for smoother integration into real-time air quality monitoring systems and adapting it to different environmental conditions and sensor setups. Collaboration with industry for practical implementation, the development of user-friendly interfaces, and continuous validation will be key to establishing D-RAKE as a leading solution for data compression in air quality monitoring, ultimately contributing to better environmental management and public health.

REFERENCES

[1] F. Zulfiryansyah, S. Syahrorini, and M. N. Habibi, "Air quality monitoring system using unmanned aerial vehicle (UAV) quadcopter type," *Procedia of Engineering and Life Science*, vol. 2, no. 2, Aug. 2022, doi: 10.21070/pels.v2i2.1244.

[2] A. Bushnag, "Air quality and climate control Arduino monitoring system using fuzzy logic for indoor environments," in *2020 International Conference on Control, Automation and Diagnosis (ICCAD)*, Oct. 2020, pp. 1–6, doi: 10.1109/ICCAD49821.2020.9260514.

[3] A. Hilary Kelechi *et al.*, "Design of a low-cost air quality monitoring system using Arduino and ThingSpeak," *Computers, Materials & Continua*, vol. 70, no. 1, pp. 151–169, 2022, doi: 10.32604/cmc.2022.019431.

[4] D. Munera, D. P. Tobon V., J. Aguirre, and N. G. Gomez, "IoT-based air quality monitoring systems for smart cities: A systematic mapping study," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 11, no. 4, p. 3470, Aug. 2021, doi: 10.11591/ijece.v11i4.pp3470-3482.

[5] R. E. Ogu, N. Chukwuchekwa, G. A. Chukwudebe, I. A. Ezenugu, and I. E. Achumba, "A robust IoT-based air quality monitoring node for multi-location deployment," *International Journal of Engineering Research & Technology (IJERT)*, vol. 11, no. 03, 2022, doi: 10.17577/IJERTV11IS030082.

[6] K. D. Purkayastha, R. K. Mishra, A. Shil, and S. N. Pradhan, "IoT based design of air quality monitoring system web server for android platform," *Wireless Personal Communications*, vol. 118, no. 4, pp. 2921–2940, Jun. 2021, doi: 10.1007/s11277-021-08162-3.

[7] S. Kaur, S. Bawa, and S. Sharma, "IoT enabled low-cost indoor air quality monitoring system with botanical solutions," in *2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, Jun. 2020, pp. 447–453, doi: 10.1109/ICRITO48877.2020.9197895.

[8] R. Lounas, D. E. Salhi, H. Mokrani, R. Djerbi, and M. T. Bennai, "Towards a smart data transmission strategy for iot monitoring systems: application to air quality monitoring," in *2019 International Conference on Theoretical and Applicative Aspects of Computer Science (ICTAACS)*, Dec. 2019, pp. 1–7, doi: 10.1109/ICTAACS48474.2019.8988119.

[9] M. W. Sari and B. Santoso, "Developing indoor air quality monitoring system using internet of things and wireless sensor network," *Jurnal Ilmiah Teknosains*, vol. 7, no. 2/Nov, pp. 13–18, Nov. 2021, doi: 10.26877/jitek.v7i2/Nov.9763.

[10] I. Lili, A. Kosta, and E. Xhina, "The use of smart devices (IoT) to monitor the air quality: a case study at the Faculty of Natural Sciences," in *Proceedings of RTA-CSIT 2023*, Tirana, Albania, 2023.

[11] I. M. Pu, "Audio compression," in *Fundamental Data Compression*, Elsevier, 2006, pp. 171–188.

[12] C. Kim and C. -C. J. Kuo, "Data compression," in *Handbook of Computer Networks*, Wiley, 2007, pp. 199–211.

[13] K. Sayood, *Introduction to data compression*. Elsevier, 2012.

[14] D. Salomon and G. Motta, *Handbook of data compression*. London: Springer London, 2010.

[15] G. Murugesan and R. Gilmary, "Compression of text files using genomic code compression algorithm," *International Journal of Engineering & Technology*, vol. 7, no. 2.31, p. 69, May 2018, doi: 10.14419/ijet.v7i2.31.13399.

[16] Y. Ji, W. Xu, and A. Deng, "A study of vessel trajectory compression based on vector data compression algorithms," Springer, 2019, pp. 473–484.

[17] D. D M, "IoT based air quality monitoring system," *International Journal for Research in Applied Science and Engineering Technology*, vol. 9, no. VIII, pp. 402–406, Aug. 2021, doi: 10.22214/ijraset.2021.37337.

[18] J. Buelvas, D. Múnera, D. P. Tobón V., J. Aguirre, and N. Gaviria, "Data quality in IoT-based air quality monitoring systems: a systematic mapping study," *Water, Air, & Soil Pollution*, vol. 234, no. 4, p. 248, Apr. 2023, doi: 10.1007/s11270-023-06127-9.

[19] A. Puscasiu *et al.*, "Indoor air quality monitoring system for healthcare facilities," in *IFMBE Proceedings*, Springer International Publishing, 2022, pp. 399–408.

[20] Z. Ahmed *et al.*, "Lossy and lossless video frame compression: a novel approach for high-temporal video data analytics," *Remote Sensing*, vol. 12, no. 6, p. 1004, Mar. 2020, doi: 10.3390/rs12061004.

[21] P. Delgosha and V. Anantharam, "Universal lossless compression of graphical data," *IEEE Transactions on Information Theory*, vol. 66, no. 11, pp. 6962–6976, Nov. 2020, doi: 10.1109/TIT.2020.2991384.

[22] A. Gopinath and M. Ravisankar, "Comparison of Lossless Data Compression Techniques," in *2020 International Conference on Inventive Computation Technologies (ICICT)*, Feb. 2020, pp. 628–633, doi: 10.1109/ICICT48043.2020.9112516.

[23] S. K. Routray, A. Javali, K. P. Sharmila, W. Semunigus, M. Pappa, and A. D. Ghosh, "Lossless compression techniques for low bandwidth networks," in *2020 3rd International Conference on Intelligent Sustainable Systems (ICISS)*, Dec. 2020, pp. 823–828, doi: 10.1109/ICISS49785.2020.9315936.

[24] Y. Im and S. Verdú, "Optimal universal lossless compression with side information," *IEEE Transactions on Information Theory*, pp. 1–1, 2024, doi: 10.1109/TIT.2018.2868053.

[25] S. K. Routray, A. Javali, A. Sahoo, W. Semunigus, and M. Pappa, "Lossless compression techniques for low bandwidth IoTs," in *2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, Oct. 2020, pp. 177–181, doi: 10.1109/I-SMAC49090.2020.9243457.

[26] M. Goyal, K. Tatwawadi, S. Chandak, and I. Ochoa, "DeepZip: lossless data compression using recurrent neural networks," in *2019 Data Compression Conference (DCC)*, Mar. 2019, pp. 575–575, doi: 10.1109/DCC.2019.00087.

[27] V. R. Joseph and S. Mak, “Supervised compression of big data,” *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 14, no. 3, pp. 217–229, Jun. 2021, doi: 10.1002/sam.11508.
[28] B. Ryabko, “Time-universal data compression,” *Algorithms*, vol. 12, no. 6, p. 116, May 2019, doi: 10.3390/a12060116.
[29] P. SY and N. Nagaraj, “Causal discovery using compression-complexity measures,” *Journal of Biomedical Informatics*, vol. 117, p. 103724, May 2021, doi: 10.1016/j.jbi.2021.103724.
[30] C. Li, J. Wang, and M. Li, “Spatiotemporal compression-transmission strategies for energy-harvesting wireless sensor networks,” *IET Communications*, vol. 13, no. 5, pp. 630–636, Mar. 2019, doi: 10.1049/iet-com.2018.5353.
[31] S. A. Abdulzahra, A. K. M. Al-Qurabat, and A. K. Idrees, “Compression-based data reduction technique for IoT sensor networks,” *Baghdad Science Journal*, vol. 18, no. 1, p. 0184, Mar. 2021, doi: 10.21123/bsj.2021.18.1.0184.
[32] G. Shrividhiya, K. S. Srujana, S. N. Kashyap, and C. Gururaj, “Robust data compression algorithm utilizing LZW framework based on huffman technique,” in *2021 International Conference on Emerging Smart Computing and Informatics (ESCI)*, Mar. 2021, pp. 234–237, doi: 10.1109/ESCI50559.2021.9396785.
[33] D. Barannik, “Stegano-compression coding in a non-equalible positional base,” in *2020 IEEE 2nd International Conference on Advanced Trends in Information Theory (ATIT)*, Nov. 2020, pp. 83–86, doi: 10.1109/ATIT50783.2020.9349328.
[34] T. S. Gunawan, Y. M. Saiful Munir, M. Kartiwi, and H. Mansor, “Design and implementation of portable outdoor air quality measurement system using Arduino,” *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 8, no. 1, p. 280, Feb. 2018, doi: 10.11591/ijece.v8i1.pp280-290.
[35] X.-G. Wu, Z.-B. Chu, X. Zheng, X.-J. Wang, and P.-L. Yang, “Sparse random projection compressive data gathering in lossy wireless sensor networks,” *Jisuanji Xuebao/Chinese Journal of Computers*, vol. 42, no. 2, pp. 388–402, 2019, doi: 10.11897/SP.J.1016.2019.00388.
[36] K. Hossain and S. Roy, “A data compression and storage optimization framework for IoT sensor data in cloud storage,” in *2018 21st International Conference of Computer and Information Technology (ICCIT)*, Dec. 2018, pp. 1–6, doi: 10.1109/ICCITECHN.2018.8631929.
[37] S. Minewaki, M. Iwahashi, H. Kobayashi, T. Yoshida, and H. Kiya, “Near lossless coding of sparse histogram images based on zero-skip quantization,” *Multimedia Tools and Applications*, Sep. 2017, doi: 10.1007/s11042-017-5082-2.
[38] P. Ramalingam, R. Thanuja, R. Bhavani, and L. Gopalakrishnan, “An efficient lossless telemetry data compression and fault analysis system using 2SMLZ and CMOW-DLNN,” *Wireless Personal Communications*, vol. 127, no. 3, pp. 2325–2345, Dec. 2022, doi: 10.1007/s11277-021-08799-0.
[39] S.-H. Hwang, K.-M. Kim, S. Kim, and J. W. Kwak, “Lossless data compression for time-series sensor data based on dynamic bit packing,” *Sensors*, vol. 23, no. 20, p. 8575, Oct. 2023, doi: 10.3390/s23208575.
[40] G. Campobello, A. Segreto, S. Zanafi, and S. Serrano, “RAKE: a simple and efficient lossless compression algorithm for the internet of things,” in *2017 25th European Signal Processing Conference (EUSIPCO)*, Aug. 2017, pp. 2581–2585, doi: 10.23919/EUSIPCO.2017.8081677.
[41] L. Sergey, G. Larisa, and G. Larisa, “Acquisition and reducing of redundancy of measuring information on the basis of national instruments reconfigurable input-output modules in technical system diagnosis,” in *2019 International Russian Automation Conference (RusAutoCon)*, Sep. 2019, pp. 1–6, doi: 10.1109/RUSAUTOCON.2019.8867805.
[42] U. Jayasankar, V. Thirumal, and D. Ponnurangam, “A survey on data compression techniques: From the perspective of data quality, coding schemes, data type and applications,” *Journal of King Saud University - Computer and Information Sciences*, vol. 33, no. 2, pp. 119–140, Feb. 2021, doi: 10.1016/j.jksuci.2018.05.006.
[43] I. Pu, “Data compression.” University of London International Programmes, London, 2004.
[44] J. López Pascual, J. C. Meléndez Rodríguez, and S. Cruz Rambaud, “The enhanced-earned value management (E-EVM) model: a proposal for the aerospace industry,” *Symmetry*, vol. 13, no. 2, p. 232, Jan. 2021, doi: 10.3390/sym13020232.

BIOGRAPHIES OF AUTHORS

Kartika Sari received the S.Kom. degree in electrical engineering from Universitas Tanjungpura, Indonesia, in 2015 and the M.Cs. degrees in degrees in Department of Computer Science from Universitas Gadjah Mada, Indonesia, in 2019, respectively. Currently, she is a lecturer at the Department of Computer System Engineering, Faculty of Mathematics and Natural Sciences, Universitas Tanjungpura Pontianak, Indonesia. She can be contacted at email: kartika.sari@siskom.untan.ac.id.

Rahmi Hidayati received the S.Kom. Department of Informatics Engineering in Universitas Islam Indonesia 2008, Indonesia and the M.Cs. degrees in Department of Computer Science from Universitas Gadjah Mada, Indonesia, in 2012, respectively. Currently, she is a lecturer at the Department of Computer System Engineering, Faculty of Mathematics and Natural Sciences, Universitas Tanjungpura Pontianak, Indonesia. She can be contacted at email: rahmihidayati@siskom.untan.ac.id.