Sensors and Materials, Vol. 34, No. 11 (2022) 4095–4111
MYU Tokyo

S & M 3099

Agile Lossless Compression Algorithm for Big Data of Solar Energy Harvesting Wireless Sensor Network

Hazem M. El-Hageen,[1,2,3*] Hani Albalawi,[1,2] Aadel M. Alatwi,[1,4]
Walaa R. Abd Elrahman,[5] and Sultan T. Mohammed Faqeh[1]

[1]Electrical Engineering Department, Faculty of Engineering, University of Tabuk,
P.O. Box 741, Tabuk 71491, Saudi Arabia
[2]Renewable Energy and Energy Efficiency Center (REEEC), University of Tabuk,
P.O. Box 741, Tabuk 71491, Saudi Arabia
[3]Egyptian Atomic Energy Authority, P.O. Box 29, Naser City, Cairo 11787, Egypt
[4]Industrial Innovation and Robotic Center (IIRC), University of Tabuk, P.O. Box 741, Tabuk 71491, Saudi Arabia
[5]Mechanical Engineering Department, Faculty of Engineering, University of Tabuk,
P.O. Box 741, Tabuk 71491, Saudi Arabia

(Received September 16, 2022; accepted November 2, 2022)

Keywords: lossless compression algorithm, solar energy, compressed time series data, wireless sensor network

Time series data are collected through most of the applications that permeate our lives today. Internet of Things (IoT) sensor data are generated through smart applications and stored in databases. Time series databases require huge storage spaces, as over time they consume a large amount of memory. In this paper, we propose an enhanced compression algorithm for time series data generated by IoT systems that monitor the production of electrical energy by solar panels. The best way to ensure that solar energy systems have high efficiency is to continuously monitor all electrical and environmental factors. However, this requires the collection of enormous quantities of data that can be used to detect defects in the generation of electric energy or in solar panels. As the data must be available for analysis, a lossless compression algorithm is needed. In addition, the compressed data must be in a format that can be queried to perform analysis operations dependent on speed; this means that the decompression of data should not be time-consuming. Our results showed the high speed of the compression process along with good compression rate (16.6%) after applying the proposed compression algorithm.

1. Introduction

Time series data are generated by various applications, such as smart electricity grid applications,(1) health-monitoring sensor systems,(2) Internet applications,(3) and Internet of Things (IoT) applications.(4–6) These applications require a huge storage capacity to store the data to be analyzed later, leading to the need to compress the data. Data compression, in addition to saving storage capacity, simplifies the data transfer and enhances the performance of time series databases.(7)

*Corresponding author: e-mail: helhageen@ut.edu.sa
https://doi.org/10.18494/SAM4106

Compression techniques fall into two main groups: lossless compression and lossy compression.(8) Lossless compression is used to retrieve the original data for various purposes.(9,10) IoT and Internet applications retrieve data for analysis,(11) making lossless compression suitable for these applications. In the case of lossy compression, some data are removed during the compression process, making it impossible to recover the original data.(12) This technique can be used to compress image, video, and audio files (e.g., JPEG, MPEG-4, and MP3 files, respectively). A large set of compression algorithms is used or directed to a specific domain. Various publications presenting time series compression algorithms have been reviewed with a focus on lossless compression(13) in the context of storing data for use in managing the maintenance operations of solar panels.

The removal of data redundancy is the basis of lossless compression techniques. Several basic compression methods for numerical data that are based on more complex methods to deal with the exponential growth of data have been developed. One of these methods is delta coding,(14) which depends on storing the difference between successive elements inside the data being compressed. The delta coding method is suitable for time series data generated by IoT systems where the recorded data, for example, temperature data, change slowly.(15) The delta coding algorithm was developed by designing the delta-of-delta encoding algorithm based on Facebook's Gorilla time series database.(16) In this case, time stamps are compressed to benefit from the fact that most time series are generated at a fixed interval; therefore, a value representing only the difference between the time series is recorded in the timestamp column.(16) Another method is the Huffman code, a lossless data compression algorithm based on the existence of frequencies of data; it is performed by building a Huffman tree.(17) To compress float data and overcome the difficulty in using all available bits even after the compression, Facebook developed the Exclusive-OR (XOR) compression algorithm.(18) In this algorithm, only different bits are stored in the memory, saving a large amount of storage space.

Pelkonen *et al.*(19) introduced a compression algorithm based on XOR compression for the data generated by monitoring Facebook systems, improving the production query latency by a factor of 70. Furthermore, the Sprintz algorithm based on machine learning was introduced to compress the time series produced by IoT sensors, aiming at bypassing the memory and latency limitations of these devices.(20)

Deep learning and machine learning methods have been used to compress data in various studies. For example, a recurrent neural network that tracks long-term dependences in time series was used to compress data.(21) In a study by Li *et al.*,(22) a method of compressing images based on convolutional networks was presented. Another example is the introduction of an image compression algorithm using a long short-term memory (LSTM) network to provide high-quality images in a minimal storage space.(23) Bidirectional LSTM (Bi-LSTM) has been used to compress and extract features from time series data collected from a power grid.(24)

Monitoring the characteristics of photovoltaics (PV) is crucial for optimization of the production process, solar energy reliability, and optimum usage, as well as for the prolonging the lifespan of their components, to lower the operation and maintenance costs.(25–27) At the Renewable Energy and Energy Efficiency Center (REEEC) of the University of Tabuk, a network of wireless sensors was set up to gather big data on electrical and environmental factors from solar power plants. On the basis of the above discussion and the need to compress time series data while maintaining the ability to retrieve them or use them in their compressed form, we propose an improved compression algorithm for time series data that are automatically stored in a database. This algorithm stores data in a format that can be easily used to perform queries and analyze the data for maintenance and monitoring operations.

2. Materials and Methods

2.1 Time series data

Rows of solar panels, inverters, battery banks, and loads are the basic units in a solar energy production system. These devices store solar-panel-generated energy. As a result, any damage or weakness (e.g., dust accumulation) will result in not only lower production but also the loss of operation of an entire row of panels due to their internal connectivity. Their continuous, effective, and efficient operation is ensured by monitoring them and taking immediate action in the case of failure. The PV solar monitoring system has been designed in order to enable real-time monitoring by modifying the measurement intervals(1–20 s), set at 5 s in this study. The structure data file (SDF) format was used to create the database. SDF files contain a compact relational database saved in the structured query language (SQL) server compact format as a preliminary step to minimize the data size.

A wireless sensor network (WSN) based on ZigBee technology was installed to collect data on electrical parameters, such as the voltage, current, and power of each solar panel, inverter, and battery bank, as well as environmental data, such as the temperature of each solar panel, the ambient temperature, the incident irradiance of sunlight, and the humidity of the solar field. The WSN was installed at the REEEC site (latitude: 28.38287N; longitude: 36.48396E; elevation: 781 m), where three identical operating solar systems with a combined capacity of 3 × 3 kW are installed. The solar farm comprises 36 (12 × 3 rows) GCL-P6/60265W (1658 × 992 mm^2) solar panels, each with 60 polycrystalline silicon solar cells with a maximum output of 265 W (8.55 A × 31 V). The station has three battery banks with a maximum voltage of 56 V per bank (4-series battery × 12 V/200 Ah) and three inverters with 60–115 V/80 A/5 kW capacities each (Fig. 1). A block diagram of the architecture of the acquisition devices, including the wireless sensors placed throughout the solar plant, is shown in Fig. 2. The measurements were recorded by sensors (see Table 1 for specifications) and include solar irradiance, humidity, rain, temperature, current, and voltage.

The output signals from the sensors were captured by the Arduino/Leonardo board with a built-in ZigBee socket, which collects and sends all data using the ZigBee WSN distributed throughout the monitoring area of the PV subsystems. The physical and medium access control layers specified in the Institute of Electrical and Electronics Engineers (IEEE) standard 802.15.4(28) serve as the foundation for the ZigBee network. A point-to-multipoint WSN (one master node to 43 slave nodes; the slave nodes cover 36 PVs, three inverters, three battery banks, and one environmental station) was installed using a 900 MHz ZigBee transceiver with TXRX modules with a long range (up to 9 miles) and a power of 250 mW. Figure 3 shows the circuits of the different WSN nodes. A multi-platform application called XCTU was used to configure, test, and simulate the ZigBee modules.

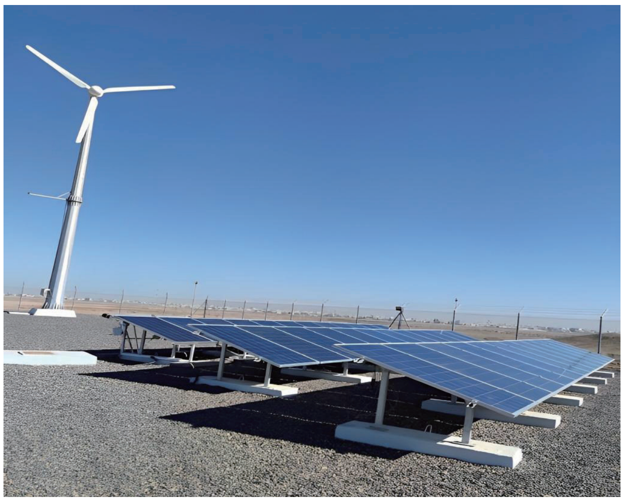


Fig. 1. (Color online) REEEC solar system with a total capacity of 9 kW.

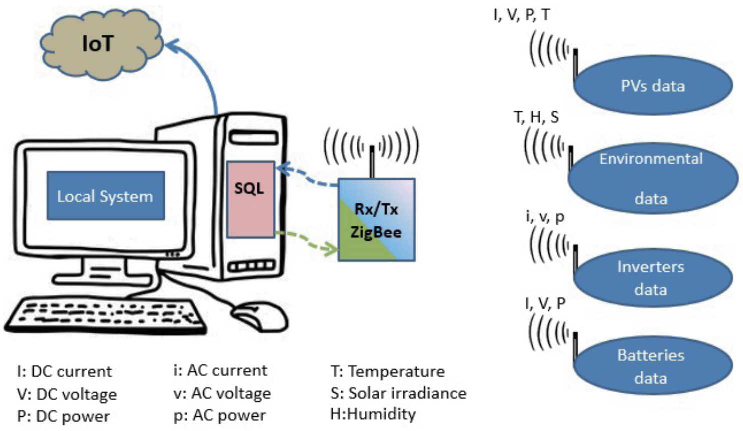


Fig. 2. (Color online) Block diagram of the WSN.

Table 1
Specifications of the sensors.

Sensor	Model	Node	Specifications
DC current	ACS725LLCTR-10AU	PVs	Isolated current sensor Sensitivity: 264 mV/A Unidirectional sensing range: 0–10 A Analog output signal
Temperature	NTC-LM393		Temperature sensing range: −25–+80 °C Analog output signal
DC current	ACS758LCB-100B	Battery banks	Isolated current sensor Sensitivity: 0.02 V/A Bidirectional sensing range: −100–100 A Analog output signal
DC voltage	Voltage divider	PVs and battery banks	Sensing range: 25–100 V Analog output signal

Table 1
(Continued) Specifications of the sensors.

Sensor	Model	Node	Specifications
AC voltage/current	PZEM-004T	Inverters	Power range: 0–9999 kW Voltage range: 80–260 VAC Current range: 0–100 A Working frequency: 45–65 Hz Serial output signal
Pyranometer	RK200-03	Environmental	Irradiance range: 0–2000 W/m^2 Sensitivity: 7–14 μVW^{-1}m^2 Operating temperature: −40–+80 °C Wavelength range: 300–3200 nm Analog output signal
Temperature and humidity	DHT11		Humidity sensing range: 20–95% Temperature sensing range: −25–+60 °C Digital output signal

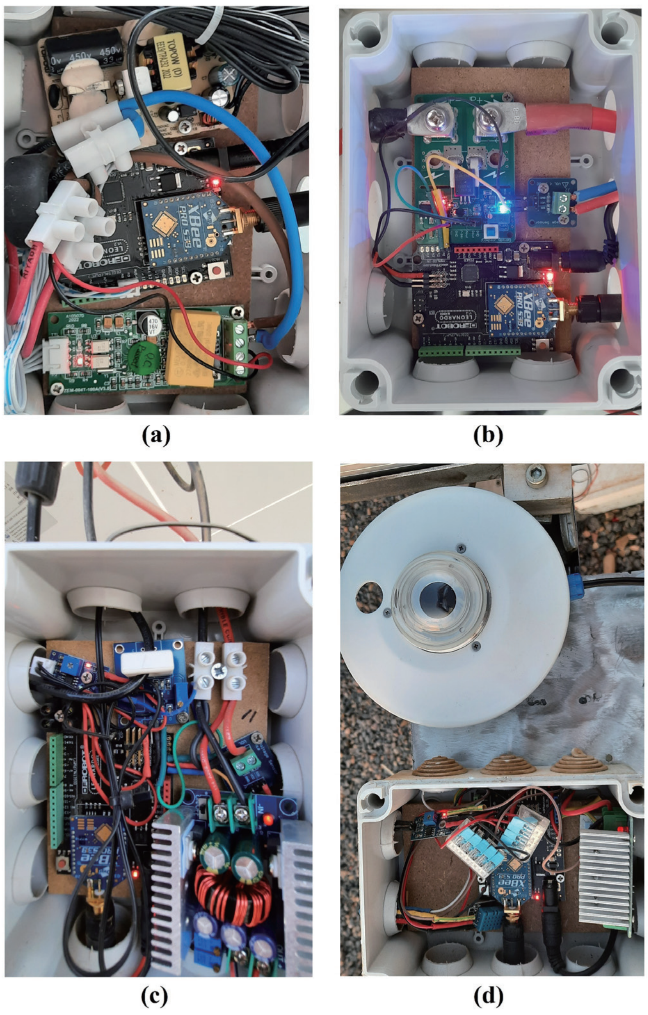


Fig. 3. (Color online) Circuit photographs of (a) inverter, (b) battery bank, (c) PV panel, and (d) environmental nodes.

Using the spectrum analyzer tool of XCTU, energy levels between the master node and the farthest PV nodes were measured in real time (Fig. 4). These signals were transmitted even at sunset when the solar panel’s output was lowest. Several channels used for transmission had signal strengths as low as −90 dB, with the lowest acceptable level being −70 dB. In this case, the transmitted signals were exposed to noise, which is redundant data and must be excluded. Under weather conditions optimal for solar energy output, the minimum power of the same channels ranged between −80 and −50 dB, ensuring that no unwanted radio noise was recorded (Fig. 5). The interval of the reception periods and the strength of the received signals for the farthest and nearest nodes were tested with the received signal strength indicator in the early afternoon (measured at 16:20–16:30; irradiance: ~1100 W/m^2), late afternoon (measured at 17:30–18:00; irradiance: ~400 W/m^2), and early evening (measured at 18:20–19:00; irradiance: <100 W/m^2). A huge amount of data was recorded during 8 months from January to the end of August 2022, while the strength of signals was measured on 26 August 2022 (Figs. 6 and 7). The synchronization between the readings from the farthest and nearest nodes was tested by calculating the average data transfer (Figs. 8 and 9).

The temperature, voltage, and current data from the solar panel were stored in the database (Table 2). These data and their changes over time are respectively shown in Figs. 10–12. The temperature changed slowly over time, while the voltage and current generated by the solar panel changed rapidly. Although the collected dataset was huge, the slow variability of these data can be used to improve the compression ratio. The dataset can be divided into three main categories: 1) repeated records (identical values) that occur when environmental parameters such as temperature and sunlight fluctuate slowly over a short period; 2) consistently trivial readings

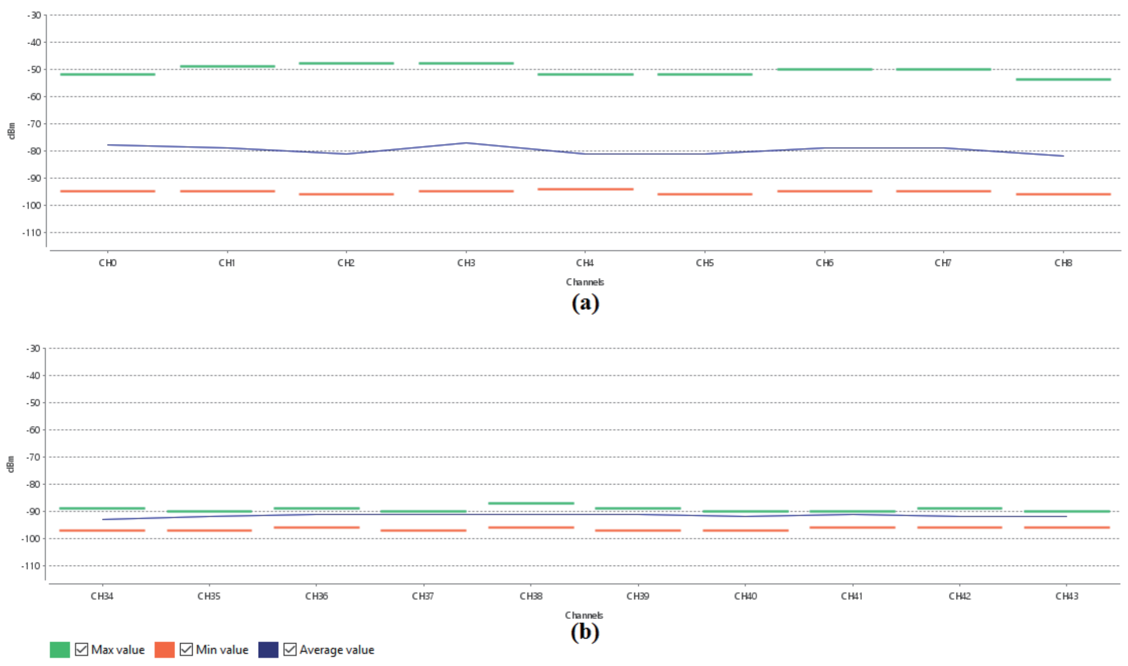
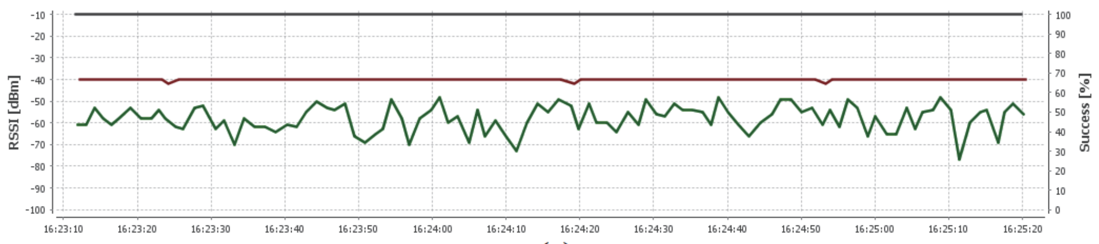
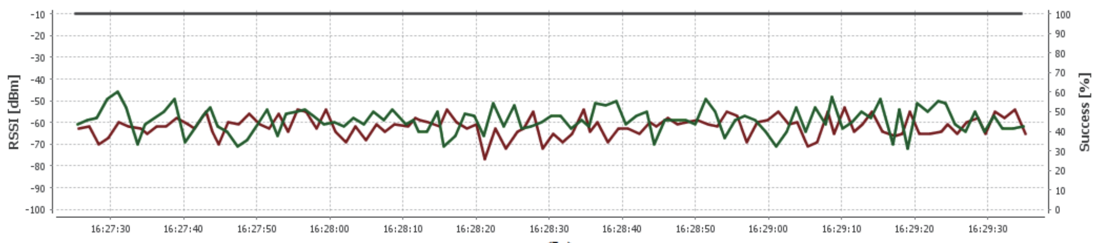


Fig. 4. (Color online) Spectrum analysis of node channels: (a) strong and (b) weak signals.



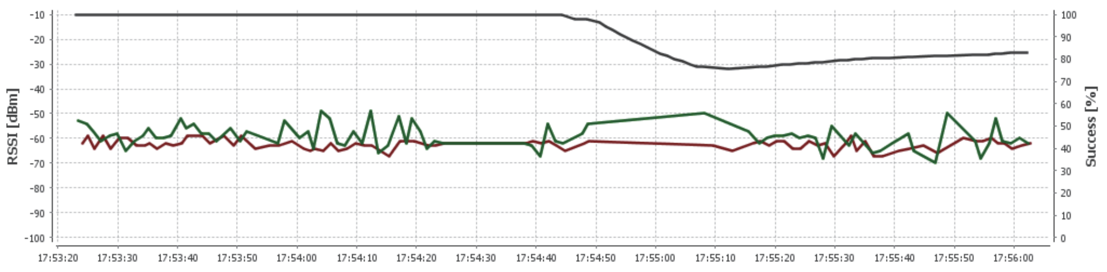
(a)



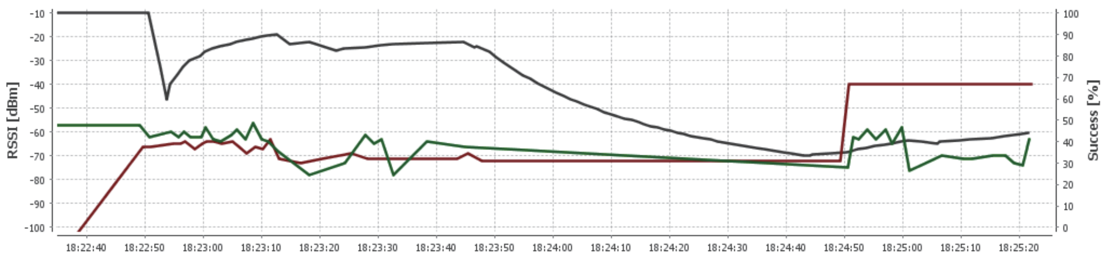
(b)

🟩 ☑ Local RSSI, the transmitted signal by the master node
🟥 ☑ Remote RSSI, the received signal from the slave node
⬛ ☑ Percentage

Fig. 5. (Color online) Radio range and received signal interval tests for (a) nearest and (b) farthest nodes in early afternoon.



(a)



(b)

🟩 ☑ Local RSSI, the transmitted signal by the master node
🟥 ☑ Remote RSSI, the received signal from the slave node
⬛ ☑ Percentage

Fig. 6. (Color online) Radio range and received signal interval tests of farthest node in (a) late afternoon and (b) early evening.

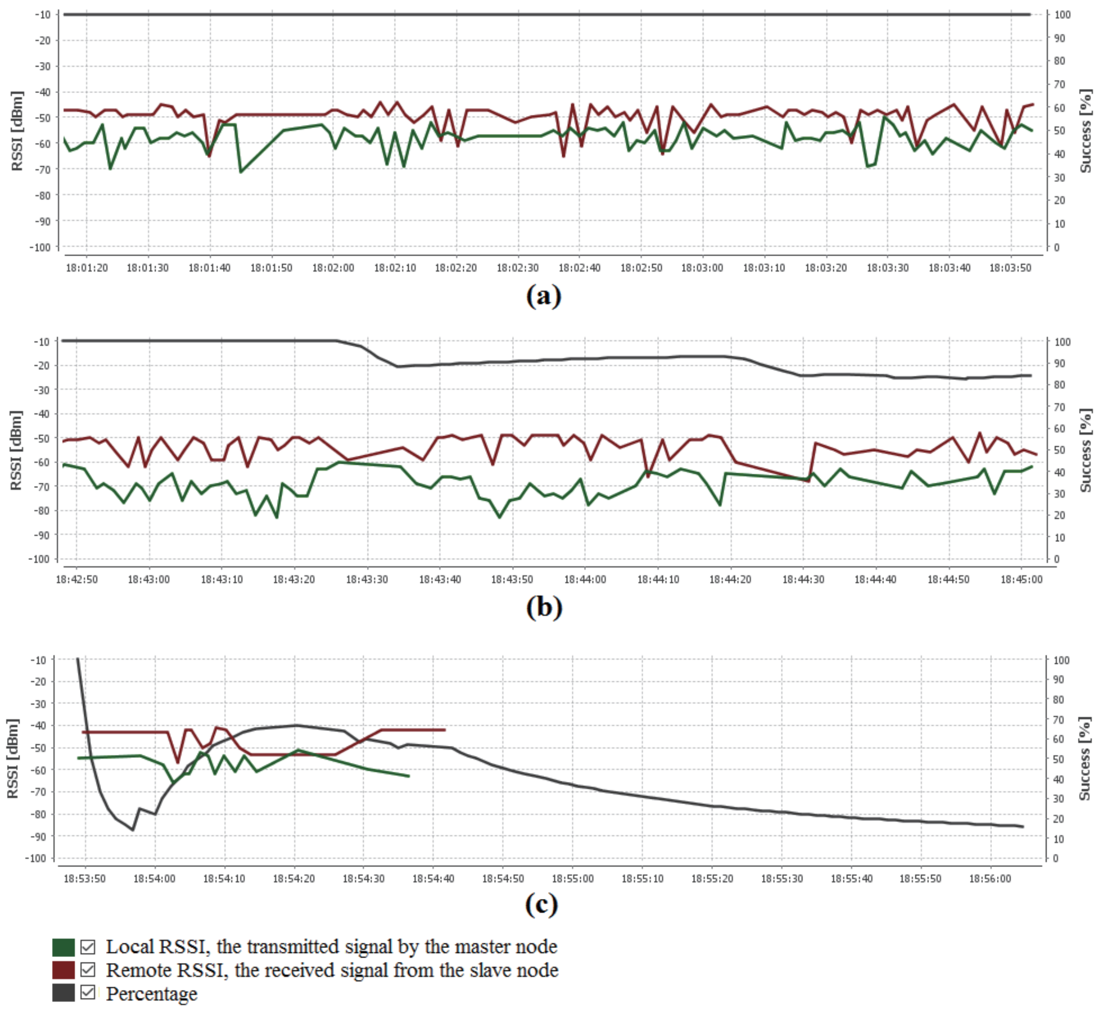


Fig. 7. (Color online) Radio range and received signal interval tests of nearest node during (a) late afternoon, (b) early evening, and (c) power blackout and dropped mode of nearest ZigBee node.

for current and voltage due to low values close to zero after sunset; and 3) false data caused by noise and distortions in the broadcast signals of the wireless transceiver nodes. The latter is an uncommon phenomenon and different from changes due to cloudy weather in that it occurs suddenly and instantly.

2.2 Measurement of compression quality

To measure the performance of the proposed compression algorithm, the compression ratio was used and is defined as

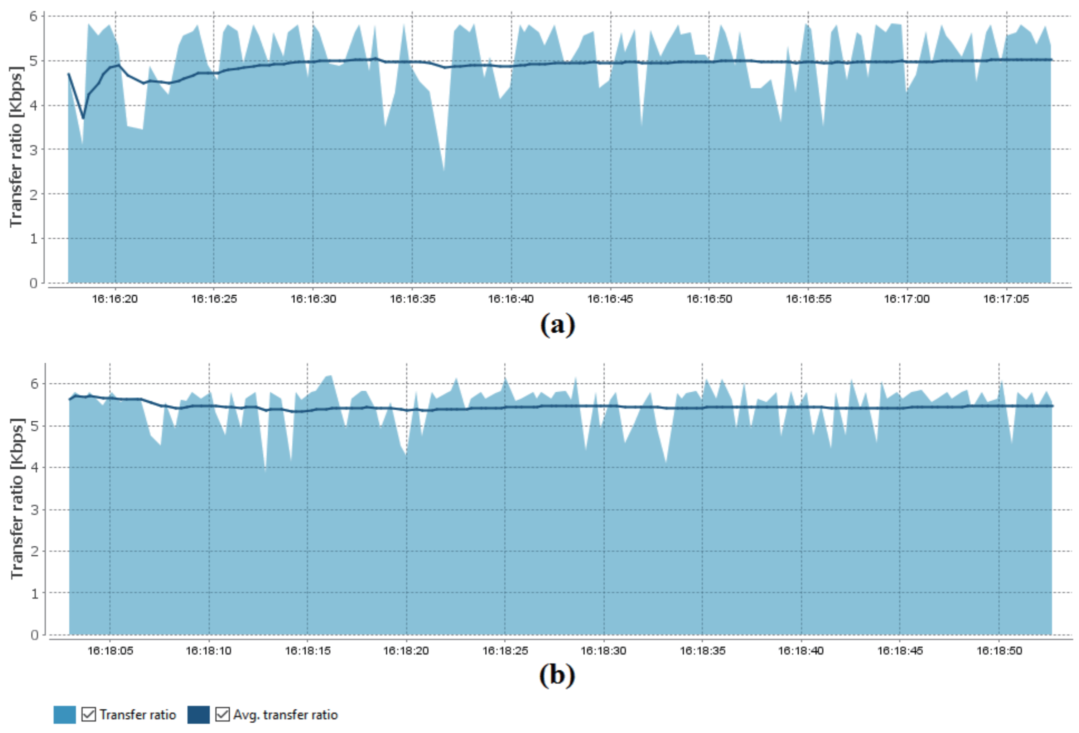


Fig. 8. (Color online) Transfer ratio of (a) farthest and (b) nearest nodes during daytime.

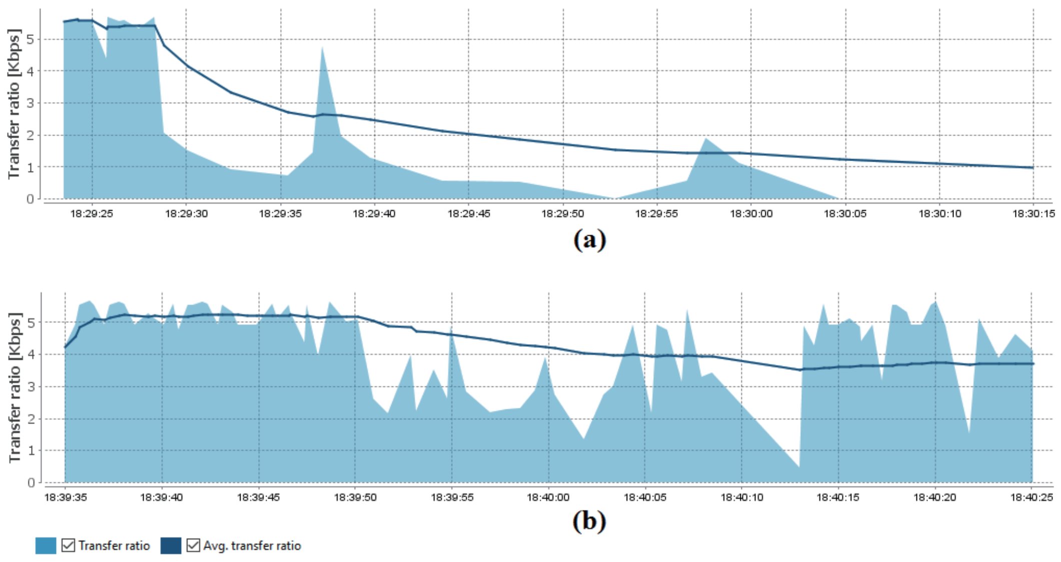


Fig. 9. (Color online) Transfer ratio of (a) farthest and (b) nearest nodes at sunset.

Table 2
Data generated from solar panel.

Time	Temperature	Voltage	Current
1/18/2022 9:10:45 AM	15	31	3
1/18/2022 9:10:50 AM	15	31	3
1/18/2022 9:10:55 AM	15	32	3
1/18/2022 9:11:00 AM	15	31	3
1/18/2022 9:11:05 AM	15	31	3
1/18/2022 9:11:10 AM	15	31	3
1/18/2022 9:11:15 AM	15	31	2
1/18/2022 9:11:20 AM	15	30	2
1/18/2022 9:11:25 AM	15	29	2
1/18/2022 9:11:30 AM	15	28	2

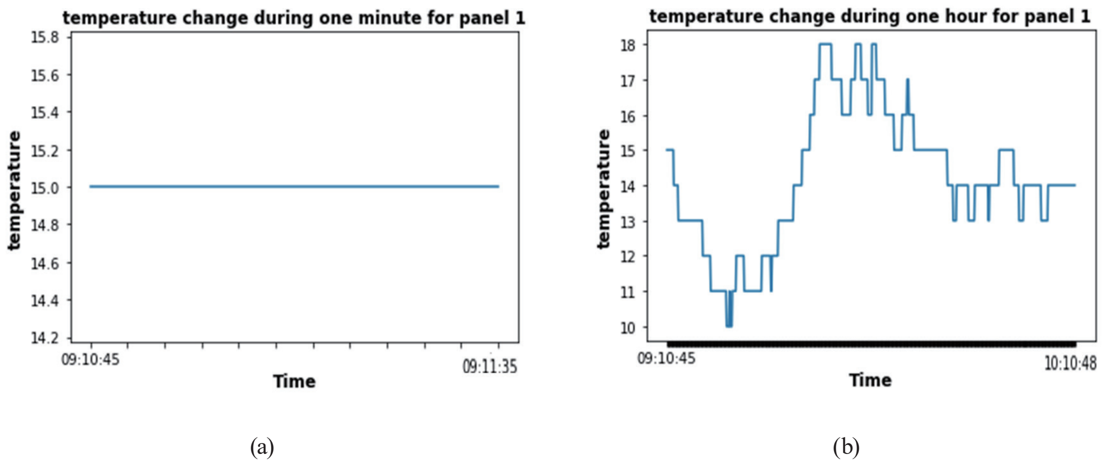


Fig. 10. (Color online) Temperature changes (a) in 1 min and (b) hourly intervals.

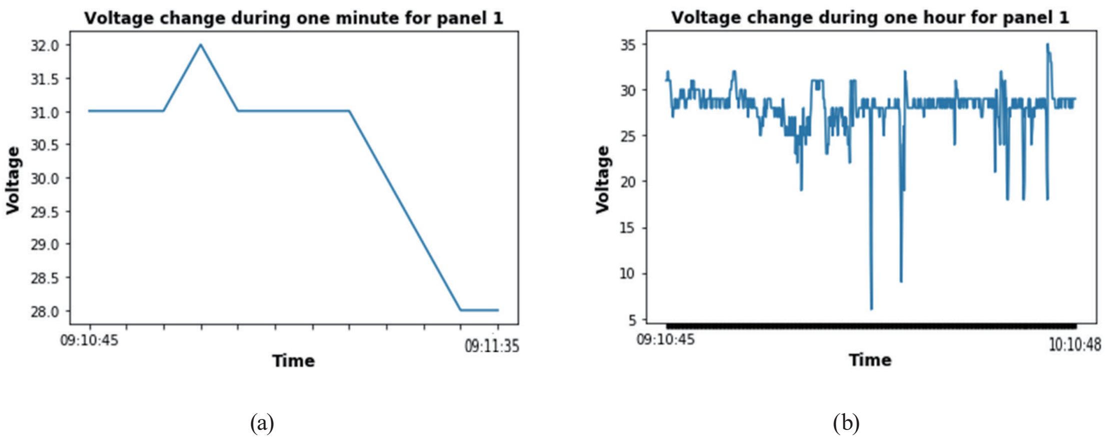
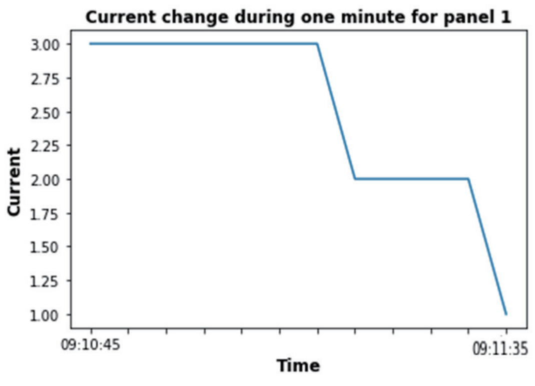
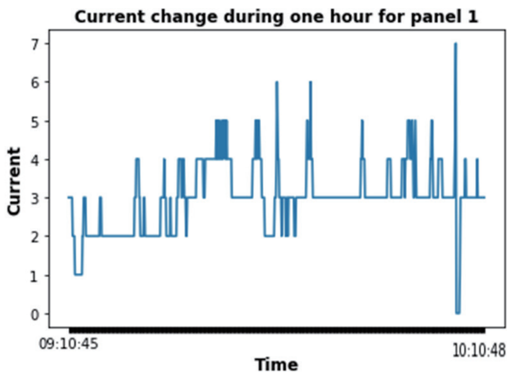


Fig. 11. (Color online) Voltage changes (a) in 1 min and (b) hourly intervals.



(a)



(b)

Fig. 12. (Color online) Current changes (a) in 1 min and (b) hourly intervals.

$$Compression\ Ratio = \frac{Original\ Size}{Compressed\ Size}. \tag{1}$$

In addition, the accuracy of the compressed data was evaluated by retrieving and decompressing the data and comparing them with the original data.

2.3 Proposed compression algorithm

The data collected from the solar panels were stored in a database and automatically compressed every 24 h. Thereafter, they were stored in separate tables and not in the original table, which is emptied every 24 h after compression. Figure 13 shows the lossless compression algorithm that manages the data of electricity production from the solar panel system. To analyze the compressed data, they were first decompressed, then retrieved and stored in the original table for use.

Owing to the importance of the data and the need to preserve all the information, lossless compression was applied to allow the data to be restored to its original form. The lossless compression used in this study required the successful elimination of different forms of data redundancy. Overall, the proposed compression algorithm is based on providing a high compression rate for the big data generated by IoT devices along with high-speed decompression to analyze the data over multiple periods.

The compression algorithm involved in the data management framework was based on redundancy, where temperature, voltage, and current values at close intervals are repeated. The proposed algorithm is divided into three phases to compress the data to the maximum extent while maintaining the ability to query the data when compressed.

The first phase is responsible for removing unwanted and outlier data since IoT devices can incorrectly record errors. For example, when the values of the electric current generated by the solar panels are recorded as zero or do not coincide with the values recorded before or after

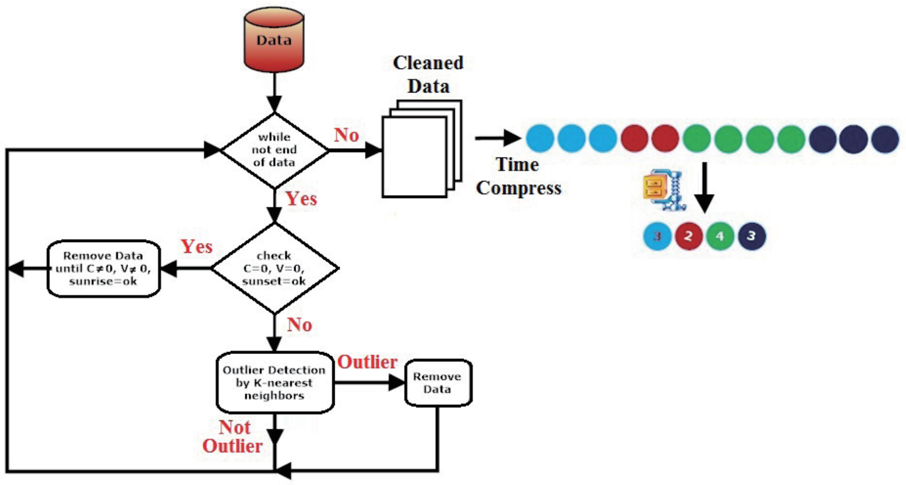

Data
while not end of data
No
Yes
Cleaned Data
Time Compress
3
2
4
3
check C=0, V=0, sunset=ok
Yes
Remove Data until C≠0, V≠0, sunrise=ok
No
Outlier Detection by K-nearest neighbors
Outlier
Remove Data
Not Outlier

Fig. 13. (Color online) Compression algorithm.

them, then they are identified as errors and are removed. Algorithm 1 illustrates this process, where the data recorded after sunset are discarded along with the errors.

The second phase of compression records only the time change between the current record and the previous record. As an example, the following four records are presented [date (YYYY/MM/DD) and time (HH:MM:SS P)]:

X1:2022/01/18; 09:10:44 AM
X2: 2022/01/18; 09:10:49 AM
X3: 2022/01/18; 09:10:54 AM
X4: 2022/01/18; 09:10:59 AM

Then, they are stored in the database by entering the time difference (measured in s) between the readings starting with the second row as follows:

X1: 2022-01-18; 09:10:44 AM
X2: 5
X3: 5
X4: 5

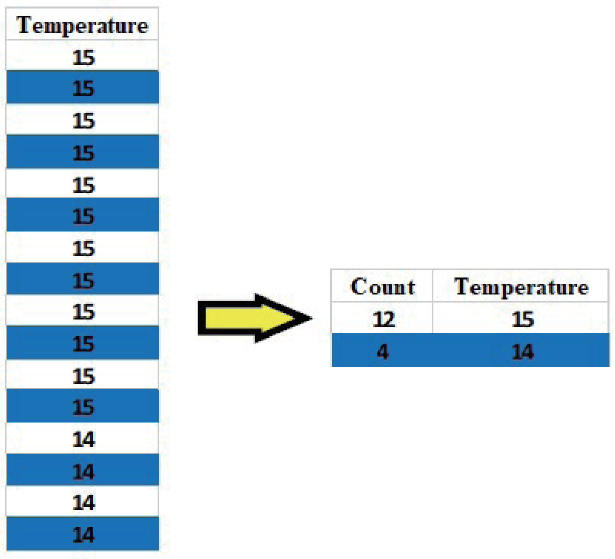
The data generated after the second phase are significantly compressed, reducing the required storage space. This also makes it possible to easily analyze the compressed data without decompressing them.

Finally, the third phase involves the compression of the temperature, voltage, and current data by collecting consecutive identical rows of the dataset. Figure 14 shows how the data were compressed in the third phase, where the number of consecutive and identical rows was calculated so that 11 rows containing a temperature of 15 ºC were compressed to one row and eight rows containing a temperature of 14 ºC were also compressed to one row.

To decompress the data, the number of iterations and their values in the compressed data were extracted, and a number of rows corresponding to these values and equal to the number of iterations was created and stored. The decompression algorithm is described in Algorithm 2,

Algorithm 1
Time series compression.

```
      Input:  List of Rows Ts (Contain the time series), D = Date and Time, T = Temperature, C = Current,
              V = Voltage
                                   Phase 1
1     Removing Unwanted and Outlier Data
                                   Phase 2
2     While is not end of Ts
3         Read current Row
4         If current Row = first Row then DateTime = Raw Value
5         else
6             DateTime = value of change between current and previous Date and Time
7     End
                                   Phase 3
8     While is not end of Ts
9         Read current Row
10        If current Row = first Row then
11            Old_D = D; Old_T = T; Old_V = V; Old_C = C
12            countDate = 1,countTemprature = 1, countVoltage = 1, countCurrent = 1
13        End
14        else
15            New_D = D; New_T = T; New_V = V; New_C = C
16            CompressData (New_D, Old_D, countDate)
17            CompressData (New_T, Old_T, countTemprature)
18            CompressData (New_V, Old_V, countVoltage)
19            CompressData (New_C, Old_C, countCurrent)
20            Old_D = New_D; Old_T = New_T; Old_V = New_V; Old_C = New_C
21        End
22    End
      Function CompressData(First, Second, count)
          If  First=Second then count= count + 1
          Else store count and first values
      End
```



Temperature
15
15
15
15
15
15
15
15
15
15
15
15
14
14
14
14

Count	Temperature
12	15
4	14

Fig. 14. (Color online) Compressed temperature data.

Algorithm 2
Time series decompression.

```
        Input:    List of compressed Data, D Rows (Count, Value), T Rows (Count, Value),
                  C Rows (Count, Value), V Rows (Count, Value)
1       DecompressData (DateTime, D)
2       DecompressData (Temperature, T)
3       DecompressData (Voltage, V)
4       DecompressData (Current, C)
        Function DecompressData (Type, Rows)
                  While is not end of Rows
                          For i = 1 to Rows. Count
                                  Store Value in Field (Rows. Type)
                          End
                  End
        End
```

where the data of date and time, temperature, voltage, and current are decompressed and stored in a separate table. The decompression process depends solely on the third phase of compression of Algorithm 1, as this is sufficient to obtain complete information about the original data to apply data analysis.

3. Results and Discussion

In this section, we discuss the performance of the proposed compression and decompression algorithms within the framework of data management in the solar power generation system. Our experiments were aimed at measuring the performance of these algorithms and were based on data recorded from solar panels for approximately 24 h.

Figure 15 illustrates the compression ratio after applying the first phase (removal of unwanted and outlier data). The compression ratio was approximately 3:1 resulting in a storage space of 120 kB with a storage savings rate of 66%.

The second phase of compression entails representing the date and time fields as time intervals to save storage space since they are stored every 5 s. The storage size of the data after applying this step is shown in Fig. 16. The compression ratio was approximately 1.3:1 after the execution of the second phase, resulting in a storage space of 91 kB with a storage savings rate of 24%.

Finally, in the third phase of the compression, successive equal rows are presented as one row. A comparison between the storage space before and that after the execution of this phase is shown in Fig. 17. The compression rate was 1.5:1, resulting in a storage space of 91 kB with a storage savings rate of 33%.

The first and second phases of compression are time-consuming, while the third phase is performed faster (9.25 s). The total time of the compression algorithm executed daily was 40.6 s, which is fast. The decompression process for daily data required only 11 s, which is a very short time and one that will be sufficient for our needs. In general, the saving in storage space amounted to about 83%, which helps users to store data without discarding it. Also, the decompression process is very fast and allows the data to be analyzed reliably. The proposed

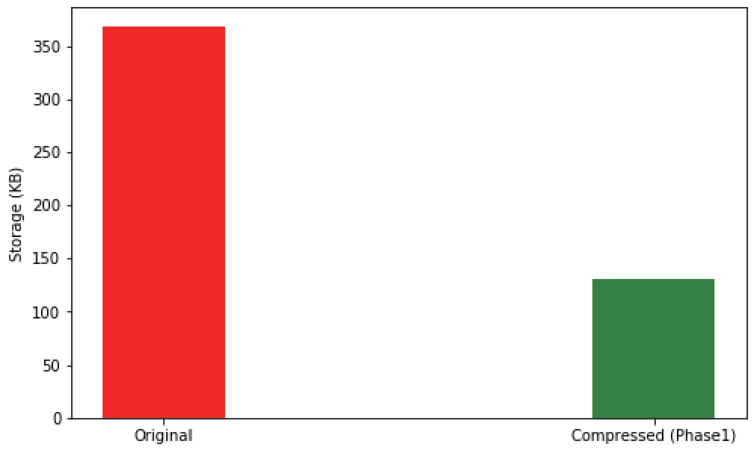


Fig. 15. (Color online) Compression ratio after first phase of compression process.

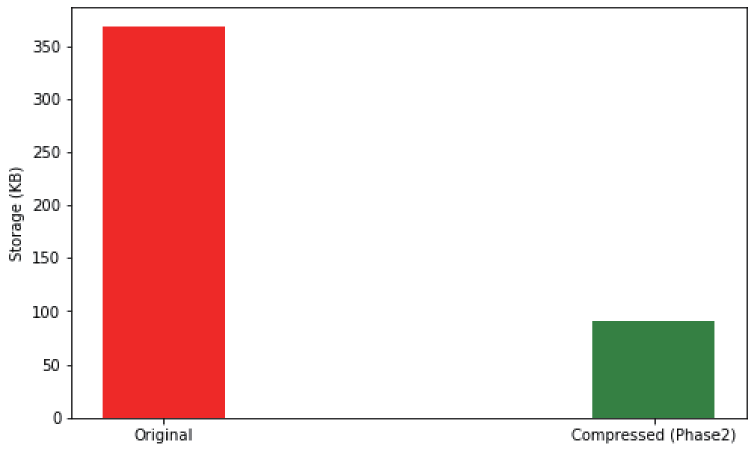


Fig. 16. (Color online) Compression ratio after second phase of compression process.

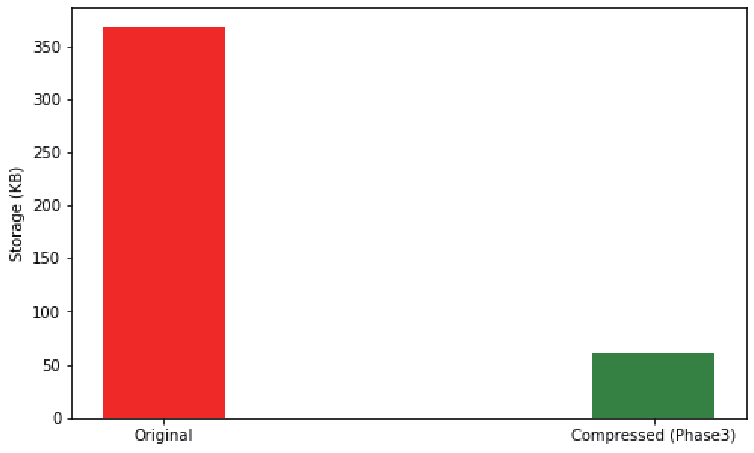


Fig. 17. (Color online) Compression ratio after third phase of compression process.

compression algorithm can be used to compress other time series data, especially those with stable values such as blood pressure measurements in intensive care. On the other hand, the proposed algorithm does not exhibit good performance in cases of rapidly changing time series data.

4. Conclusion

The compression of time series data obtained from IoT systems is essential, as the resulting databases are stored at high rates and used for decision-making. In this study, we proposed an algorithm for the compression of data with properties related to an IoT system that monitors solar panels and their production levels. This algorithm is used to compress the data daily, meaning that the compression process must be fast. Additionally, the data resulting from the compression process must be accessible for queries without decompressing them.

The proposed algorithm is divided into three phases. The first phase removes unwanted and outlier data. The second phase arranges date and time data and converts them to values of seconds that represent the difference between the current and previous rows, saving a large amount of storage space. Finally, the third phase exploits the properties of the data and collects them in a single row, further significantly reducing the storage space. It is sufficient to apply the decompression algorithm only to the data generated from the third phase of the compression algorithm, as the resulting data can be easily processed to perform queries.

This method reduces the time required to decompress data, especially time series data, which usually have a very large volume. The results also showed that the compression ratio after the second stage was 16.6%. Also, the daily decompression time was in seconds, and the reverse decompression time was less than 30 s for a full day of data.

Acknowledgments

The authors received financial support for this work from the Deanship of Scientific Research (DSR), University of Tabuk, Tabuk, Saudi Arabia (grant number: S-1442-0193).

Conflict of interest

The authors have no conflicts of interest to declare relevant to the content of this article.

References

1 F. Eichinger, D. Pathmaperuma, H. Vogt, and E. Müller: Computational Intelligent Data Analysis for Sustainable Development, T. Yu, N. Chawla, and S. Simoff, Eds. (Chapman and Hall/CRC, 2013) 1st ed., Chap. 7.
2 Yue-Lin Hsieh, Yue-Da Hsieh, and W. Wang: Sens. Mater. **34** (2022) 2791. https://doi.org/10.18494/SAM3917
3 F. Martinez-Alvarez, A. Troncoso, J. Reyes, M. Martínez-Ballesteros, and J. C. Riquelme: Comput. Intell. Neurosci. **2017** (2017) 1. https://doi.org/10.1155/2017/9361749
4 R. Kumar, P. Kumar, and Y. Kumar: Procedia Comput. Sci. **167** (2020) 373. https://doi.org/10.1016/j.procs.2020.03.240
5 A. Saidani, X. Jianwen, and D. Mansouri: Wireless Commun. Mobile Comput. **2020** (2020). https://doi.org/10.1155/2020/8824954

6 K. Totsu, M. Moriyama, H. Watanabe, T. Kikuta, M. Hemmi, M. Shoji, T. Yoshida, and M. Tatsuta: Sens. Mater. **31** (2019) 2555. https://doi.org/10.18494/SAM.2019.2390
7 T. Schlegl, S. Schlegl, D. Tomaselli, N. West, and J. Deuse: Adv. Eng. Inform. **52** (2022) 101629. https://doi.org/10.1016/j.aei.2022.101629
8 D. Salomon: A Concise Introduction to Data Compression (Springer, London, 2008) 1st ed., Chap. 2.
9 Y. Dua, V. Kumar, and R. S. Singh: J. Parallel, and Distributed Comp. **150** (2021) 60. https://doi.org/10.1016/j.jpdc.2020.12.004.
10 K. L. Ketshabetswe, A. M. Zungeru, B. Mtengi, and C. K. Lebekwe: IEEE Access **10** (2021) 136872. https://doi.org/10.1109/ACCESS.2021.3116311
11 R. Vestergaard, Q. Zhang, M. Sipo, and D. E. Lucani: IEEE Internet Things J. **8** (2021) 17568. https://doi.org/10.1109/JIOT.2021.3081868
12 J. D. Correa, A. S. Pinto, and C. Montez: Internet Things **19** (2022) 100516. https://doi.org/10.1016/j.iot.2022.100516.
13 A. Gómez-Brandón, J. R. Paramá, K. Villalobos, A. Illarramendi, and N. R. Brisaboa: Comput. Ind. **132** (2021) 103503. https://doi.org/10.1016/j.compind.2021.103503
14 S. Banerjee and G. K. Singh: Biomed. Signal Process. Control **79** (2023) 104127. https://doi.org/10.1016/j.bspc.2022.104127
15 J. Azar, A. Makhoul, R. Couturier, and J. Demerjian: Neurocomp. **398** (2020) 222. https://doi.org/10.1016/j.neucom.2020.02.097
16 T. Suel: Encyclopedia of Big Data Technologies, S. Sakr and A. Zomaya, Eds. (Springer, Cham, 2018) p. 1. https://doi.org/10.1007/978-3-319-63962-8_63-1
17 D. A. Huffman: Proc. **40** (1952) 1098.
18 M. Burtscher and P. Ratanaworabhan: IEEE Trans. Comput. **58** (2009) 18. https://doi.org/10.1109/TC.2008.131
19 T. Pelkonen, S. Franklin, J. Teller, P. Cavallaro, Q. Huang, J. Meza, and K. Veeraraghavan: Proc. VLDB Endowment **8** (2015) 1816. https://doi.org/10.14778/2824032.2824078
20 D. Blalock, S. Madden, and J. Guttag: Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. **2** (2018) 1. https://doi.org/10.1145/3264903
21 K. Bascol, R. Emonet, E. Fromont, and J. M. Odobez: Proc. Structural, Syntactic, and Statistical Pattern Recognition (S&SSPR 2016) Conf. 10029 (2019) 427. https://doi.org/10.1007/978-3-319-49055-7_38
22 M. Li, W. Zuo, S. Gu, D. Zhao, and D. Zhang: Proc. 2018 IEEE/CVF Conf. Computer Vision and Pattern Recognition (2018) 3214. https://doi.org/10.1109/CVPR.2018.00339
23 G. Toderici, S. M. O'Malley, S. J. Hwang, D. Vincent, D. Minnen, S. Baluja, M. Covell, and R. Sukthankar: Proc. 4th Int. Conf. Learning Representations (ICLR) (2016). https://doi.org/10.48550/arXiv.1511.06085
24 Z. Ma, H. Zhu, Z. He, Y. Lu, and F. Song: Sensors J. **22** (2022) 5331. https://doi.org/10.3390/s22145331
25 X. F. Wu, C. Y. Yang, W. C. Han, and Z. R. Pan: Alexandria Eng. J. **61** (2022) 2775. https://doi.org/10.1016/j.aej.2021.08.003
26 S. R. Madeti: Renewable Energy Focus **41** (2022) 160. https://doi.org/10.1016/j.ref.2022.03.001
27 R. K. Pachauri, O. P. Mahela, B. Khan, A. Kumar, S Agarwal, and H. H. Alhelou, and J. Bai: Comput. Electr. Eng. **92** (2021) 107175. https://doi.org/10.1016/j.compeleceng.2021.107175
28 N. Choudhury, R. Matam, M. Mukherjee, and J. Lloret: IEEE Access **8** (2020) 41936. https://doi.org/10.1109/ACCESS.2020.2976654