IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 10, 2021, accepted September 27, 2021, date of publication September 29, 2021, date of current version October 12, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3116311

Data Compression Algorithms for Wireless Sensor Networks: A Review and Comparison

KELEADILE LUCIA KETSHABETSWE[1], (Member, IEEE), ADAMU MURTALA ZUNGERU[1], (Senior Member, IEEE), BOKANI MTENGI[1], (Member, IEEE), CASPAR K. LEBEKWE[1], (Member, IEEE), AND S. R. S. PRABAHARAN[2], (Member, IEEE)
[1]Department of Electrical, Computer and Telecommunications Engineering, Botswana International University of Science and Technology, Palapye, Botswana
[2]Directorate of Research, SRM Institute of Science and Technology, Kattankulathur, Chengalpattu, Chennai, Tamil Nadu 603203, India

Corresponding author: Adamu Murtala Zungeru (zungerum@biust.ac.bw)

This work was supported by the Office of Research, Development, and Innovation (ORDI) of Botswana International University of Science and Technology under Grant S00297.

ABSTRACT Energy consumption has risen to be a bottleneck in wireless sensor networks. This is caused by the challenges faced by these networks due to their tiny sensor nodes that have limited memory storage, small battery capacity, limited processing capability, and bandwidth. Data compression has been used to reduce energy consumption and improve network lifetime, as it reduces data size before it can be forwarded from the sensing node to the sink node in the network. In this paper, a survey and comparison of currently available data compression techniques in wireless sensor networks are conducted. Suitable sets of criteria are defined to classify existing data compression algorithms. An adaptive lossless data compression algorithm (ALDC) is analyzed through MATLAB coding and simulation from the reviewed data compression techniques. The analysis aims to discover strategies that can be used to reduce the amount of data further before it is transmitted. From this analysis, it was discovered that encoding residue samples, rather than raw data samples, reduced the bitstream from 112 bits to a range of 30 to 36 bits depending on the sample block sizes. The average length of data samples to be passed to the encoder was minimized from the original 14 bits per symbol to 1.125 bits per symbol. This demonstrated a 0.875 code efficiency or redundancy. It resulted in an energy saving of 67.8% to 73.2%. This work further proposes a data compression algorithm that encodes the residue samples with fewer bits than the *ALDC* algorithm. The algorithm reduced the bitstream to 26 bits. The average length of the code is equal to the entropy of the data samples, demonstrating zero redundancy and an improved energy saving of 76.8% compared to *ALDC*. The proposed algorithm, therefore, shows improved energy efficiency through data compression.

INDEX TERMS Compression ratio, data aggregation, data compression, wireless sensor networks.

NOMENCLATURE

ADC	Analogue to Digital Converter.
ALDC	Adaptive Lossless Data Compression.
ALEC	Adaptive Lossless Entropy Compression.
BLE	Low Power Bluetooth.
CodeID	Code Decision Identifier.
CoC	Computational complexity.
CR	Compression ratio.
CS	Compressed/Compressive Sensing.
DSC	Distributed Source Coding.
DSM	Distributed Source Modelling.
DTC	Distributed Transform Coding.
FELACS	Fast and Efficient Lossless Adaptive Compression Scheme.
ID	Identification.
iid	independent and identically distributed.
IoP	Internet of People.
IoT	Internet of Things.
IP	Internet Protocol.
JPEG	Joint Photographic Experts Group.
LEC	Lossless Entropy Compression.
LoRa	Long Range.
LZW	Lempel Ziv.
MATLAB	Matrix Laboratory.

The associate editor coordinating the review of this manuscript and approving it for publication was Chi-Yuan Chen.

MP3	MPEG Audio Layer 3.
MPEG	Moving Pictures Experts Group.
MSTP	Minimum Spanning Tree Projection.
NB-IoT	Narrow Band Internet of Things.
QoS	Quality of Service.
RIP	Restricted Isometry Property.
RSS	Received Signal Strength.
SNR	Signal to Noise Ratio.
SR	Sampling ratio.
S-LZW	LZW for Sensor Nodes.
TTL	Time To Live.
WAN	Wide Area Network.
WSN	Wireless Sensor Network.
WSS	Wireless Sensor System.

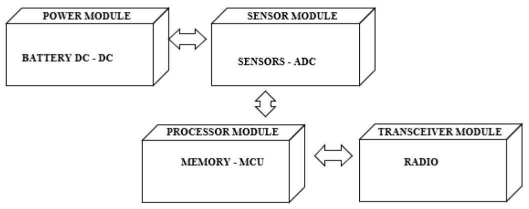
I. INTRODUCTION

Large-scale data collection for many applications can be made possible by deploying wireless sensor networks (WSNs) through continuous monitoring. Healthcare and well-being monitoring, surveillance, environmental monitoring, precise and intelligent agriculture, seismic, industrial, and structural monitoring are examples of application areas covered in wireless sensor networks [1]–[5].

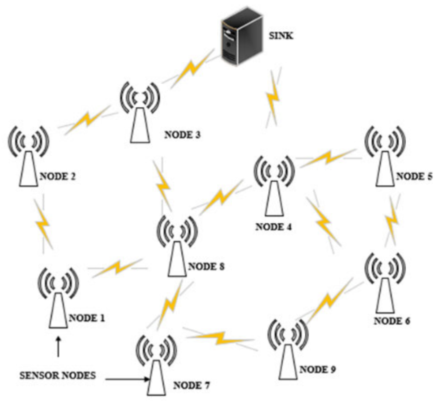
WSNs are created by combining wireless sensor nodes with wireless networks. The sensor nodes independently organize themselves and can randomly be deployed in large numbers over a field of interest. They can monitor events within the field and use radio communication to report events' information to a base point or sink node. Each node is housed with a sensor, battery, processor unit, communication unit, and memory unit. With the advent of technology development, the sensor nodes have since been reduced in size to tiny elements that suffer limitations in terms of memory, battery size, bandwidth, and processing capability. Typical structures of a wireless sensor node and a wireless sensor network are demonstrated in Fig. 1a and 1b.

Communication in a wireless sensor network is made possible by applying short-distance, low-power wireless communication technologies such as Bluetooth, Zigbee, Wireless Hart, and many others. The *WSN* lifetime thus strongly depends on the lifetime of the sensor node's battery. Since the battery is very small, strategic management of power in a *WSN* is of primary concern and needs to be applied to ensure a prolonged network lifetime [2].

Energy consumption in a *WSN* is experienced during sensing, processing, and communication of data. Receiving and data transmission consumes more energy than data processing. A data management structure is needed to resolve any data conflicts that may occur during data collection from different sensors. It is vital to minimize the loss of data during this process. Low power sensor batteries cannot be practically replaced or re-charged since the sensors are deployed in huge numbers and sometimes inaccessible. This raises the need to monitor energy consumption or management and devise strategic techniques for saving energy in *WSNs*.



(a).



(b).

FIGURE 1. (a) A wireless sensor node structure. (b) A wireless sensor network.

This paper intends to address this energy consumption issue which threatens the *WSN* life span. The approach taken by this work to address this energy consumption challenge is to reduce the size of data before it is transmitted and to employ efficient means of transmitting this data across the network and eventually to its destination or sink. This can be achieved through data compression, the application of efficient data aggregation techniques, and efficient routing techniques.

With data aggregation, data packets are combined through different ways of routing. This is achieved by manipulating some extracted features and statistics of sets of data collected from sensor nodes like the minimum, maximum and/or mean and then forwarding them to the sink. For efficient data aggregation, a routing algorithm and data compression techniques are needed [6], [7]. An efficient data compression algorithm should reduce the size of the data and compress data using fewer resources. The data user characteristics must be considered when compressing data. Irrelevant data can be eliminated depending on the ability of the user to perceive or use such data [8].

Data compression has proved to be an efficient energy-saving scheme that reduces the amount of data to be sent across a network. Fig. 2 demonstrates a data compression technique that is formed by two algorithms, whereby one takes a certain input *A* that a generated *Ac* can represent with lesser bits. The other algorithm works on the compressed *Ac* and produces a reconstruction algorithm *B*. Different structures of data and characteristics of the user can be used to

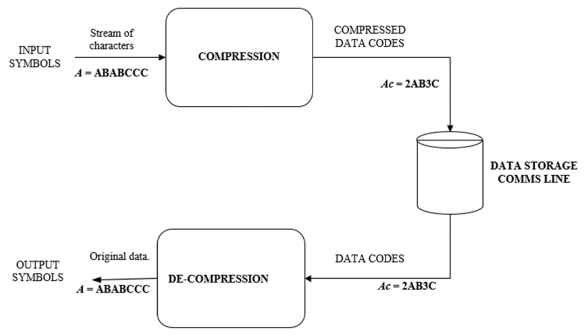


FIGURE 2. A data compression model.

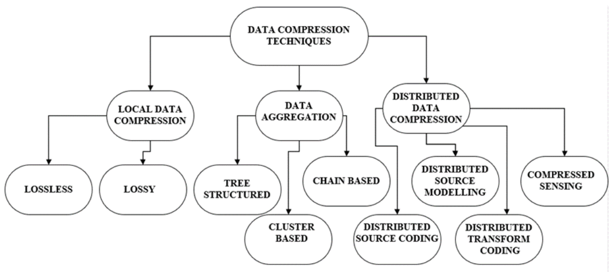

DATA COMPRESSION TECHNIQUES
LOCAL DATA COMPRESSION
DATA AGGREGATION
DISTRIBUTED DATA COMPRESSION
LOSSLESS
LOSSY
TREE STRUCTURED
CHAIN BASED
CLUSTER BASED
DISTRIBUTED SOURCE MODELLING
COMPRESSED SENSING
DISTRIBUTED SOURCE CODING
DISTRIBUTED TRANSFORM CODING

FIGURE 4. Classification of data compression techniques.

achieve data compression [8]. In Fig. 2, a 7-bit stream of data is reduced to 5-bit data that can be easily stored or transmitted over a communication line.

Data compression eliminates redundant data, and as a result, less energy is required to transmit the compressed data. Fig. 3 below illustrates the relationship between data aggregation, data compression, and routing algorithms.

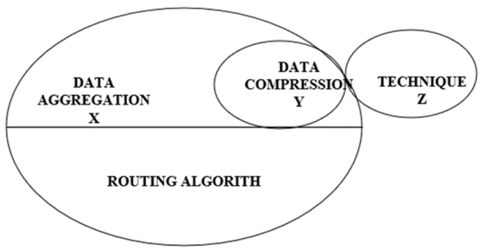


FIGURE 3. Data compression and data aggregation - The relationship.

Data aggregation techniques are commonly experienced in densely populated sensor networks where routing algorithms are needed due to the multi-hop architecture of the network. According to Fig. 3, the process of data aggregation, as illustrated by set X, sits on top of the routing algorithm. The process is performed by extracting minimum, maximum, and mean values of aggregated sensor data [8]. Data communication is, as a result, minimized. The data aggregation technique can lead to loss of the structure of the original data, which can be addressed by applying data compression techniques as indicated by set Y of Fig. 3. The compression techniques are an extension of data aggregation techniques and distribute the compression algorithm across the network. Set Z independently performs compression at every local node and does not need a routing protocol or a densely populated network. The architecture of their sensor network is sparse. Therefore, data compression algorithms can be categorized into local data compression algorithms, data aggregation, and distributed data compression algorithms. Fig. 4 illustrates this classification. The breakdown of this categorization is discussed in Section IV.

Data compression is performed in two stages of Modelling and Coding. Modelling is the initial stage where information that pertains to any redundancy in the data is extracted and described as a model. A model is referred to as static if its parameters do not change with the changing characteristics of the data. It is considered to be adaptive if its parameters change with the changing characteristics of the data. In the Coding phase, the model is compared to the data, and their difference is encoded in binary. This difference is known as the residual and can be represented in a sequence that is encoded with fewer bits than the original data.

Surveys of data compression techniques in wireless sensor networks have been carried out by researchers in this field. The contributions of this research work are as follows:

- Investigating existing data compression algorithms and classifying them according to features that define their application. The classes proposed are Local data compressions, data aggregation, and distributed data compression algorithms.
- A survey and comparison of some local data compression algorithms (*ALDC, LEC, TMT, FELACS*) and some distributed data compression algorithms (*DSC, DSM, CS, IMAGE based*) based on compression ratio, energy saving, minimized transmission, processing complexity, and net energy saving.
- Analysis and simulation of *ALDC* algorithm and proposing a data compression algorithm that improved on energy efficiency in *ALDC*. The improvement was achieved by minimizing the number of bits used to encode residue samples in *ALDC* before being forwarded to the encoder.

The rest of the Sections are as follows: Section II of the work presents the factors necessary to achieve data compression. Different data compression techniques are categorized and discussed in section III. Section IV defines some performance measures that are used to analyze selected local data compression algorithms and some distributed data compression algorithms. The proposed algorithm is discussed in section V, and the analysis and conclusions of the results are presented in section VI.

II. FACTORS TO CONSIDER IN DATA COMPRESSION

Factors that are required for data compression are based on the various ranges of wireless sensor networks applications. Some applications can tolerate a certain level of latency and or data loss, while some may not. Generally, requirements of data compression include;

A. COMPLEXITY IN PROCESSING AND MEMORY CAPACITY

Typical processing speeds for wireless sensor nodes range around 4 – 8MHz with instruction memories of 128kB and data memories of 4kB. This calls for the design of less complex compression algorithms with limited code size. It is desirable to design algorithms that perform most of the data processing at the sink instead of performing compression at individual nodes. In this case, sensors with lower processing performance are considered more efficient in data compression [6], [8], [9]. Less complex processing algorithms save both power and processing costs [6].

B. DATA EXCHANGE

A significant amount of energy used by sensor nodes is consumed during data transmission and reception. Redundant data communicated between nodes is, as a result, removed by some compression algorithms. The algorithms are designed to perform more processing and minimum transmission. This increases data processing at both the transmitter and receiver [8].

C. REDUNDANT DATA ACQUISITION

Duplicate or redundant data may occur during collection, transmission and saving of data from overlapped sensor regions covered by the nodes. Redundancy can be discovered and exploited by compression algorithms. Exchange of information between nodes is used to establish sensing schedules with minimal rate of observation.

D. RELIABILITY

In communication, measurement and spatial redundancy can be exploited to improve data reliability.

E. SCALABILITY

A data compression algorithm must be able to grow or scale with the size of the network.

F. THE COMPRESSION ALONG ROUTES

Data is traditionally compressed at the source nodes and decompressed at the sink. Data is made available at forwarding nodes for compression along routes to allow changes and processing or on-route compression. Low-performance sensor nodes can handle less complex compression while forwarding nodes that are powered by mains handle processing or compression that is more powerful.

G. ROBUSTNESS

Compression techniques must function adequately even at the occurrence of a node or link failure in the network. This calls for a trade-off between energy efficiency and robustness as redundant deployment will be needed to tolerate the sensor nodes and or link failures.

Some requirements of data compression are dependent upon the application of the wireless sensor network. Some applications can be for tracking, while others are for monitoring. Tracking applications include vehicle tracking, enemy tracking in wars, animal tracking in agriculture or wildlife management, and human tracking in-home automated applications. Requirements for practical *WSN* applications include:

H. SECURITY

Some wireless sensor networks applications demand a certain level of security, creating a conflict with data compression. Some security protocols may require encryption of sensor data before communicating the data, and decryption of the data only takes place at the sink. Data compression, on the other hand, may occur at forwarding nodes to maximize energy efficiency. Therefore, data compression algorithms and security protocols should be designed in consideration of one another to avoid this conflict [10].

I. REAL-TIME

Intelligent and healthcare applications are some examples of real-time data *WSN* applications. They demand that compression must be done one sample at a time, limiting the compression ratio. Exploiting spatial correlation is still possible, though [8]. It is essential to know the statistical structure of measured datasets. For tracking applications that normally require real-time operations, the algorithm's speed should be taken into consideration during implementation. For monitoring applications, several data types exist that can be divided into either variance or source. Data variance can either be smooth or non-smooth, which is suitable for entropy compression algorithms and can give high compression ratios. The data can be a single data type for every sensor node or multiple data types. For tracking applications, more than two sensors are joined in one module. The sensors are of the same kind and produce similar data types. Therefore, the design for compression algorithms for this kind of application demands prior knowledge of the statistical nature of sensor data. On the other hand, for monitoring applications, one node can be joined to various types of sensors to produce diverse types of data [5].

J. CONSCIOUSNESS OF QUALITY OF SERVICE (QoS)

The size of data at the destination and its perfect resemblance to the original data is referred to in wireless sensor networks as the quality of information (*QoI*) or *QoS*. It is dependent on the *WSN* application and is usually not easy to maintain because of data compression approximations and redundant data removal [8].

III. CLASSIFICATION OF DATA COMPRESSION TECHNIQUES AND RELATED WORKS

Classification of data compression techniques is based on certain features that define the application for which they are intended to be used. This work classifies these techniques into *Local data compression*, *Data aggregation,* and *Distributed data compression* techniques, as shown in Fig. 4. This section also presents recent research work on different algorithms that have been developed for *WSNs* under the other classes of data compression techniques. Different sub-categories that fall under these classes are discussed under each class in the sub-sections below.

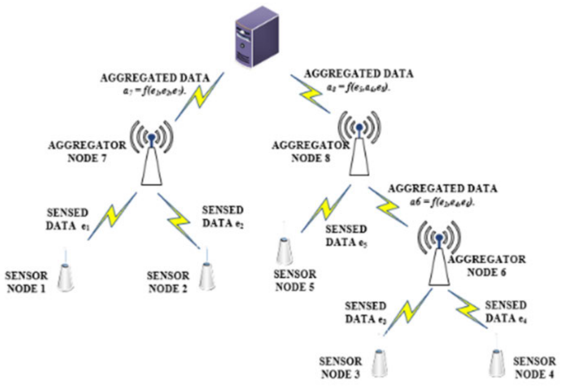


FIGURE 5. A tree structured data aggregation scheme.

A. DATA AGGREGATION COMPRESSION TECHNIQUES

Some applications require only a summary of data collected from sensor nodes. A selected set of sensor nodes are tasked with gathering and fusing sensed data from intermediate nodes. The selected aggregation nodes collect sensed data from immediate nodes, apply aggregating techniques, derive the minimums, maximums, and average values of the data. The fused data, which is reduced in size, is then sent to the sink. The original sensed data cannot be recovered from the compressed aggregated data. Hence the data aggregation compression technique is irrecoverable [9].

Work in [9] presented data aggregation techniques as simply applying in-network data computation for *WSNs*' data and transmission compression. This operation significantly minimizes the volume of sensed data that is sent to the sink. The aggregated data is, however, irrecoverable. In their research on data aggregation strategies, authors in [11] investigated efficient means of selecting aggregation nodes to minimize data communication in *WSNs*. Authors in [8] identified advantages of data aggregation as dependent on the range between sources of aggregated data as opposed to the range between the sink and the sources and the amount of the summarized data compared to the raw or original data. This led to consideration of the aggregation structure and discovering optimal ways of fusing data as discovered by [12], [13]. The data aggregation structures are reported to determine the performance of the protocols applied.

Four categories of data aggregation techniques are presented in this section.

1) TREE STRUCTURED DATA AGGREGATION TECHNIQUES

The wireless network is structured into a tree-like architecture, and data fusion is performed at forwarding nodes. The technique intends to achieve optimal joint strategies for aggregation of data and formation of routing trees, as illustrated by Fig. 5. In the figure, sensed or extracted data is denoted by e_a, which is sent by every individual node a, A forwarding or aggregation node k fuses its own sensed data e_k with the data received from its child nodes with a function of aggregation f('.',').

Data compression algorithms that fall under this category are:

a: ENERGY-AWARE DISTRIBUTED HEURISTIC APPROACH (EADAT) [14]

The root of the tree, which is the sink, identifies the source node, the intermediate nodes, the sensor node's status and residual energy, and the number of hops to the sink by broadcasting a control message. Individual sensor nodes establish a countdown timer when they receive the control message on their first encounter. The timer is activated when the communication reaches an idle state. An intermediate node with the shortest route to the sink and shows more residual energy is then selected as a parent node by the sensor node. When the counting reaches zero, the control message is updated and broadcasted to nearby nodes. To ensure that all sensor nodes are added to the tree, the operation is performed repeatedly. The structure of the tree is thus built, and to maintain it, a threshold of the remaining energy is used. The remaining energy of sensor nodes is compared with this threshold and if it is less, a help broadcast message is sent by the sensor node to its child nodes, and one of the child nodes trades places with the sensor node, maintaining the tree structure.

11). Tree based Tiny Aggregation: It is an aggregation approach that is designed particularly for low power *WSNs* and monitoring applications that use TinyOs.

111). Power Efficient Data gathering and Aggregation Protocol (PEDAP): It reduces the entire energy consumed by sensor nodes by aggregating sensed data from the different nodes using link costs to calculate minimum spanning tree as

$$\mathrm{a}_{\mathrm{i,j}}(\mathrm{n}) = P_{cct}.2n + P_{amp}.n.(r\ (i,j))^2. \tag{1}$$

The energy link cost for transmission of n bits from nodes i to j is denoted by $a_{i,j}(n)$, while the level of power dissipated by the transceiver circuit for every bit is represented by P_{cct}. The level of power consumed by the transmitter amplifier for every bit is P_{amp}, while $r_{(i,j)}$ is the range between nodes i and j. Calculation of the spanning tree is achieved by use of Prim's algorithm [15], which then forward the data packets through the tree edges to the destination node [10].

A variant of *PEDAP,* Power-aware *PEDAP (PA-PEDAP),* considers the remaining energy of the sensor nodes.

$$\text{Link costs are modified by}\,\frac{\mathrm{a}_{\mathrm{i,j}}(n)}{P_{\mathrm{i}}} \tag{2}$$

where Pi represents the normalized remaining energy of node i relative to the initial energy. Therefore, sensor nodes that incur higher link costs are not included in the spanning tree and load balancing across the sensor nodes is achieved.

Authors in [14] developed an algorithm that builds and maintains a data aggregation tree structure in sensor networks. It is however less robust and requires regular querying. To reduce the entire energy consumed by sensor nodes, work in [16] presented an algorithm that aggregates sensed data from the different nodes to improve network lifetime through load balancing [9]. For low power, WSNs, and monitoring applications, [17] designed a TinyOS approach. Its variant Power-aware *PEDAP (PA-PEDAP)* considered the remaining energy of the sensor nodes. Even though *PA-PEDAP* outperforms *LEACH* and *PEGASIS,* unfortunately, it operates from a centralized point and depends on the general knowledge of the network.

2) CHAIN BASED DATA AGGREGATION TECHNIQUES

In these techniques, sensors send collected data to their neighboring nodes, and a long chain is formed that connects the sensors. All sensor nodes in the chain become aggregation nodes, excluding those at the end of the chain. An example of these techniques is Power Efficient Data Gathering Protocol for Sensor Information Systems (*PEGASIS*). The global knowledge of the network is assumed to be known by every sensor node. The node that is farthest away from the sink initiates the formation of the chain. On every cycle of chain formation, its neighboring node, which is nearest to the sink, is selected to succeed the sensor node in the chain. Every sensor node receives sensed data from its neighbor after the formation of the chain. The data is aggregated with its own sensed data and forwarded along the chain to its successor. The cycle is repeated until all the aggregated data in the network reaches the sink. An approach that constructs chains in the same manner as *PEGASIS* but can lower energy consumption by reducing the range D between sensor nodes that are adjacent in the chain inserts a fresh sensor node into the chain at every iteration. The addition of a new node increases the minimum amount of $\sum D^2$ All sensor nodes in Fig. 6 are aggregator nodes, except for sensor node 1 at the end of the chain and is responsible for collecting sensed data. of the current chain. Repetition of this operation allows all sensor nodes to be added to the chain. A concept that illustrates a chain-based data aggregation scheme is illustrated in Fig. 6.

A chain-based data aggregation technique, *PEGASIS* uses chain formation that does not guarantee lower energy consumption. To cater to this, authors in [18] proposed an approach that constructs chains in the same manner as PEGASIS but can lower energy consumption by reducing the range between sensor nodes adjacent to the chain. This scheme, however, adds complexity in time for the number of sensor nodes used [19].

3) CLUSTER BASED DATA AGGREGATION TECHNIQUES

Sensor nodes are grouped into clusters, and cluster heads are chosen to aggregate sensed data from sensor nodes for each cluster, as illustrated in Fig. 7. Cluster heads then send the aggregated data to the sink, either through single hop

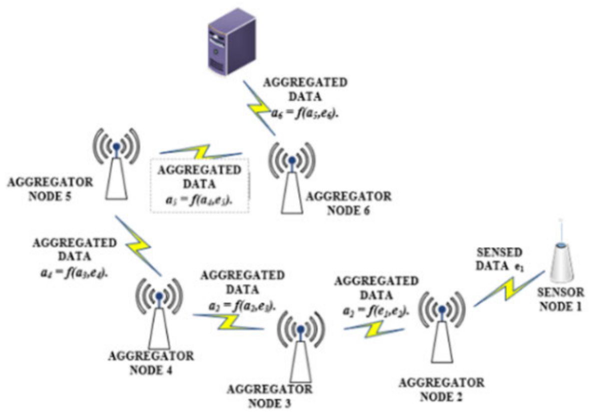

AGGREGATED DATA $a_6 = f(a_5,e_6)$.
AGGREGATOR NODE 5
AGGREGATED DATA $a_5 = f(a_4,e_5)$.
AGGREGATOR NODE 6
AGGREGATED DATA $a_4 = f(a_3,e_4)$.
AGGREGATED DATA $a_3 = f(a_2,e_3)$.
AGGREGATED DATA $a_2 = f(e_1,e_2)$.
SENSED DATA e_1
SENSOR NODE 1
AGGREGATOR NODE 4
AGGREGATOR NODE 3
AGGREGATOR NODE 2

FIGURE 6. A chain-based data aggregation structure.

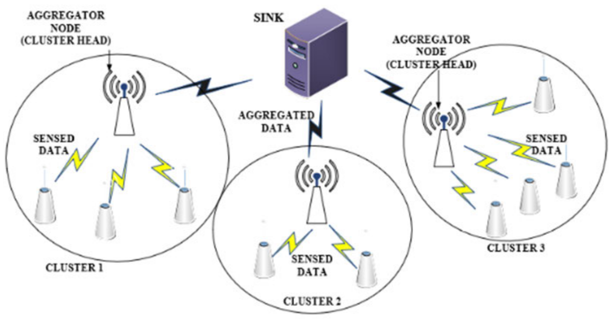


FIGURE 7. A cluster-based data aggregation structure.

or multi-hop transmission. Low Energy Adaptive Clustering Hierarchy (*LEACH*) proposed by [20] is one example of cluster-based data aggregation techniques. It operates in a setup phase which organizes clustering in the network, and the steady-state phase, where the selected cluster heads handle data aggregation. Another example of these techniques is Hybrid Energy-Efficient Distributed clustering (*HEED)* [21] which selects cluster heads according to the sensor nodes' closeness to neighboring nodes and their remaining energy added together.

Low Energy Adaptive Clustering Hierarchy (*LEACH*) proposed by [20] and Hybrid Energy-Efficient Distributed clustering (*HEED*) [21] are some examples of cluster-based data aggregation techniques. Even though *HEED* improves network lifetime more than *LEACH* does, it requires a number of power levels at sensor nodes, which is undesirable [8].

4) SECTOR BASED DATA AGGREGATION TECHNIQUES

These are aggregation techniques that gather more information from the original data than what their counterparts can gather. Authors in [22] proposed a Semantic/Spatial Correlation aware Tree (*SCT*) technique, which is a sector-based data aggregation technique. It considers a *WSN* that forms a circle with a radius R, and the sink at the network center. The network is divided into h rings, with similar widths of R/h. Individual rings are subdivided into sectors of equal sizes.

It is assumed that all sensor nodes are uniformly distributed across the network, with each sector having almost the same number of sensor nodes. An aggregation node is appointed to gather and aggregate sensed data from other sensor nodes from every sector. Individual aggregation nodes in the i^{th} tree are connected to their parent aggregation nodes in the $(i-1)^{th}$ ring along the shortest path. This creates a tree, and then individual aggregation nodes can forward their data to the sink. It is assumed in this scheme that all sensor nodes are aware of their geographical locations [9]

To satisfy the desired quality of data and to maintain aggregation error below a set threshold, authors in [23] and [24] proposed algorithms that maximize the volume of data captured at the sinks at the expense of energy, delay, and data flow constraints. On the other hand, [25] proposed an Application Independent Data Aggregation (*AIDA*) scheme that adaptively aggregates data to manage congestion and attain reliability from one end of the network to another. The approach, however, is too complex for sensor nodes that are already resource-constrained.

5) QUALITY OF SERVICE (QoS) BASED AGGREGATION TECHNIQUES

Application Independent Data Aggregation (AIDA) is a QoS scheme that adaptively aggregates data to manage congestion and attain reliability from one end of the network to another, even under heavy traffic. These are useful to satisfy the desired quality of data and to maintain aggregation error below a set threshold. Some QoS algorithms maximize the volume of data captured at the sinks at the expense of energy, delay, and data flow constraints [23] and [24]. Authors in [26] presented a data aggregation compression scheme that combines energy awareness and QoS awareness but is restricted to linear compression protocols. For sensor nodes that are limited in resources, this technique is computationally intensive.

B. LOCAL DATA COMPRESSION TECHNIQUES

Local data compression schemes exploit the sampled sensor data's temporal correlation and compress data locally at each sensor node. These can either be '*Lossless*' or '*Lossy*.' Lossless data compression techniques are used to compress text, strings, or programs, while Lossy compression schemes compress images, video, and audio files [2].

1) STRING-BASED (TEXT) COMPRESSION TECHNIQUES

These techniques see sensed data as a string of characters. Data compression algorithms that can compress textual data are then applied to the sensed data. Text compression techniques fall under lossless data techniques where the accuracy of sensed data is usually critical for some applications like environmental monitoring, health and wellbeing monitoring, and many others.

Several compression algorithms for text data have been developed to support lossless compression. Work by [8] introduced dictionary-based compression algorithms that compress different types of data. Their strategy is best suited where structural limitations constrain patterns that frequently appear to a small-sized set of the entire possible patterns. A well-known example of a lossless text data compression algorithm that is based on the creation of dictionaries as data is read from the input stream is Lempel-Ziv-Welch (*LZW*) [27], [28], [29]. Unlike static-based schemes, these schemes' code length is dynamic. However, [30] and [31] discovered that the algorithms have high memory and computational requirements and alternatively introduced adapted versions e,g Sensor-Lempel-Ziv-Welch (*S-LZW*) [30]. A variant of *S-LZW* introduced a mini cache into the dictionary, hence *S-LZW-MC(mini-cache).* Utilization of the mini cache reduces the processing time and improves the compression ratio.

A proposal by [32] improved a version developed by [4] named *LEC*, an algorithm that outperformed *S-LZW* and the first kind of lossless compression. They introduced static Huffman coding in sensor nodes. Its modification was developed by [2], which was a classical adaptive version of the Huffman coding. Due to its processing complexity, a simple compression algorithm that uses a median predictor and an *LEC* compression table was proposed by [32]. A prediction coding-based approach that discovered temporal correlation found in sensed data, resulting in predictive data compression. Its modified version improved the coding efficiency by discovering an optimal interval that allows a higher compression ratio.

To overcome the challenge of searching for the optimal intervals that gives the desired compression ratio, a heuristic approach was developed by [33] and is adaptively processed at the destination node. Even though the *LEC* algorithm outperforms other algorithms, its setback of not being able to adapt to changes in sensed data was addressed by 'Adaptive Lossless Data Compression algorithm (*ALDC*) proposed by [2]. A fast and efficient Lossless Adaptive Compression Scheme (*FELACS)* [3] was also developed to address challenges faced by most of these algorithms. Based on Golomb-Rice coding, *FELACS* encodes data much faster than static Huffman, adaptive Huffman, and Arithmetic coding schemes.

2) IMAGE COMPRESSION TECHNIQUES

This is an approach that generally supports lossy compression and is commonly applied to video and image signals. It transforms raw data into sets of coefficients with suitable basis functions that can recover the destination's original signal. Usually, only a few samples are enough to recover the estimated raw signal with minimal error. Image-based compression takes advantage of the correlations of the statistics of the data and the structure of the network [8]. It organizes a *WSN* in a hierarchical structure and perceives sensed data as an image that is made up of several pixels. Elements of a matrix whose values are small pixels that create a picture or image can be used to model the image. Vital information can be deduced from the picture in the frequency domain through the matrix wavelet transformation and only the vital information of the picture can be kept. This reduces the size of the picture to an acceptable and considerable level. Loss of part of the sensed data may be realized during compression. Hence the technique is referred to as lossy [9]. Spatial and temporal summaries of the sensed data are then extracted using wavelet transformation. The size of sensed data can further be reduced when the spatial and temporal correlation in the sensed data is high [9].

Image compression algorithms employ the basic components of image coding, which are classified into first and second-generation coding. Under the first-generation image compression techniques, commonly known examples are Joint Photographic Experts Group (*JPEG*), Embedded Zerotree Wavelet (*EZW*), Set-Partitioning in Hierarchical Trees (*SPIHT*) and Embedded Block Coding with Optimized Truncation (*EBCOT*). Examples of second-generation image compression schemes include pyramidal coding, directional decomposition-based coding, segmentation-based coding, and vector quantization.

DIMENSIONS [34] is a framework of image-based compression techniques. This framework reduces the sensor nodes' data communication size and supports various sensed data resolutions for querying by users through the exploitation of the wavelet transformation and process of quantizing. It organizes the network into many levels with blocks at each level containing four blocks in a lower level. Cluster heads are appointed at each block of the levels to gather sensed data from their four blocks and compress the data to capture their summarized spatial correlation. Different resolution exists for the data that is stored at different levels. Finer resolution is found in the data stored at lower levels while coarse resolution exists at higher level storages. This allows more detailed data queries to be made from the cluster head at lower levels since the size of sensed data is minimized. Taking advantage of temporal correlation in the data and the knowledge deduced from the signal characteristics, individual sensor nodes reduce the size. The *DIMENSIONS* framework further compresses data from lower-level cluster heads by adopting a Huffman decoder and a dequantization module. The main compression technique that is used in *DIMENSIONS* is Discreet Wavelet Transform (*DWT*). Another framework of the image-based compression is the multi-resolution compression and query (*MRCQ*) framework [35]. It aims at establishing multi-resolution summaries of sensed data in the network by organizing sensor nodes in a hierarchy. Lower resolution summaries are sent to the sink while higher resolution summaries remain in the network and can be queried. *MRCQ* suffers less processing costs than *DIMENSIONS* and can be implemented on simple sensor platforms [9].

Authors in [36] presented a survey of image compression algorithms to discover the most suitable algorithm for *WSNs*. They illustrated the basic components of image coding, which is classified into first and second-generation coding. Results indicated that the second-generation compression schemes require more complex image compressing processes than the first-generation techniques and introduce lossy compression. Discreet Cosine Transform (*DCT*) and Discreet Wavelet Transform (*DWT*) are performed in the first-generation image compression and *DWT* outperforms *DCT* because *DCT* reduces the recovered image's quality. *SPIHT* stood out among all techniques as the most suitable for *WSNs* regarding compression efficiency and memory requirements. Work in [34] presented a framework that can minimize the sensed data and regularly report to the sink by sensor nodes. However, it may incur higher processing costs due to executing Huffman encoding approach, 3D-DWT, and quantization.

C. DISTRIBUTED DATA COMPRESSION TECHNIQUES

These techniques take advantage of the high spatial correlation of data from fixed sensor nodes in dense networks.

1) DISTRIBUTED SOURCE CODING (DSC)

DSC falls under distributed data compression techniques. It relieves the sensor nodes of the computational work and assigns it to the sink. It uses the spatial correlation in the data readings between neighboring sensors. Utilizing the Slepian-Wolf theorem, sensed data is compressed inside the network, confirming that encoding of two or more correlated data streams can be independent. At the same time, its decoding is joint at the sink at a rate that is equal to the entropy of both streams [9]. The sensors do not transmit data with one another but forward their correlated outputs to the sink for joint reconstruction [8]. The processing work is handled by the sink rather than the sensor nodes. Spatial correlation between intermediate sensor node readings is exploited, and the sensor nodes do not exchange information. Fig. 8 illustrates this phenomenon where two correlated data streams *A* and *B* are compressed using statistical knowledge of the data only and not the real value of the sensed data.

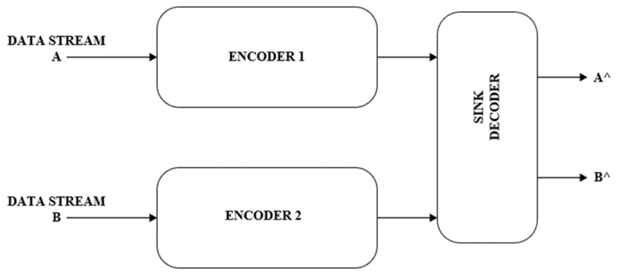


FIGURE 8. DSC based on slepian wolf theorem.

Data stream *A* is received by Encoder *1* and forwarded as a coded message to the sink. Each character of *A* is encoded by *Ra* number of bits. In the same manner, data stream *B* is transmitted to Encoder *2,* and each character of *B* is encoded by *Rb* number of bits. After receiving the coded messages, the decoder at the sink generates predictions of the original data streams *A* and *B* as two *n* vectors *A^* and *B^*. For an admissible system, lossless data compression of A and B can be achieved for larger values of n.

Authors in [37] demonstrated how compression efficiency can be achieved at a central point by compressing individual sensors' data in a distributed pattern with statistical information from other sensors rather than with the actual sensor data values. Reference [38] introduced lossy distributed compression and presented theoretical results. Extending the rate-distortion of the Slepian-Wolf theory, [39] provided a characterization tool for communication that is needed to attain a set distortion for a highly spatially correlated data network. Authors in [40] presented practical DSC techniques that involved centralized collecting, tracking correlated data, and developing code. The technique demonstrated significant power saving in several sensor modes or applications. The localized, distributed collecting, tracking, and code development version was introduced by [41]. Although it optimizes allocation and transmission of rate distortion, it becomes ineffective for small-scale neighboring nodes. It underperforms practically because it tends to avoid routing issues and is considerate of static link capacity. *DSC* compression techniques are limited by the need for prior knowledge of correlations of data at each sensor. They are also non-scalable and not robust.

2) DISTRIBUTED SOURCE MODELLING (DSM)

DSM is a distributed data compression scheme that intends to discover a function that is compatible with a set of readings collected from a given group of sensors. It uses parametric and non-parametric modelling depending on whether the data collected is treated as a random process estimated using the mean and variance of a known observed sensed data. With parametric modelling, less or no prior knowledge of the nature of the sensed data is required, making the technique to be considered robust [6]. Fig. 9 illustrates the concept of parametric modeling.

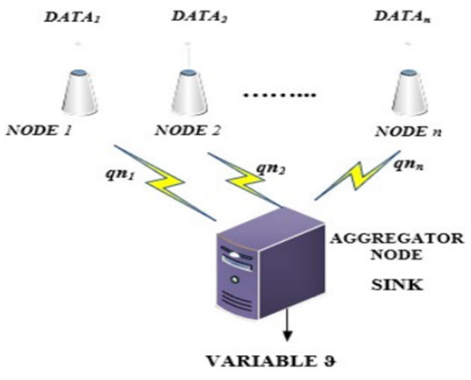


FIGURE 9. **Parametric modelling [6].**

In the figure, sensor readings $data_1$, $data_2$ up to $data_n$ are collected from a group of sensors and are transmitted to the sink, which estimates a representation of incoming data named *variable* ϑ. Quantities of real values are sent along with distortion, under constraints like noise. A quantization process is applied and produces quantized data qn_1, qn_2 up to qn_n before transmission to the sink. The data communication over the network can be represented as $data_n = \theta + No$. Here, sensed a random variable $data_n$ means data, and its unknown expectation value is θ, while No represents noise. An estimated value ϑ from the sink gives the least error between θ and its quantized data. Quantization data compression and parametric modelling are applied to sensor nodes and the sink, resulting in minimized transmitted data and energy saving.

Authors in [42] presented an application of a *DSM*, non-parametric estimation, which was founded on a non-parametric kernel-based approach. In this application, the focus was on an algorithm that trained every single sensor node to reach a working objective that shows less risks. The technique however suffered a heavy communication load that increased as the number of neighboring nodes increased. Many nodes could not be trained all at the same time. In 2010, [43] improved the rule of updating the network to resolve the limitations. The communication load was minimized from being quadratic to linear and asynchronous updates were allowed [6].

3) COMPRESSIVE SENSING (CS)

A distributed data compression technique was proposed to handle a larger number of samples that necessitated computation of all transform coefficients despite the fact that all, except those that exceed the initial samples, would be discarded. This means that, for m large number of samples, all transform coefficients $\Theta(m)$ must be processed by the encoder even though all, except F, will be discarded. This is based on the condition that $m \gg F$. In *CS*, the number of sensors required to transmit data depends on the sparsity of the signal [30], [6] and [34] which arises due to the spatiotemporal correlation between the sensor data. The nature of computation in *CS* is asymmetric and hence makes it suitable for compression of data in *WSN*. The theory of *CS* is that if a signal can be sparsely represented on one basis, then reconstruction of the signal using a fewer number of projections on to a second basis showing incoherence with the first one is possible. Instead of sampling an F-sparse signal m times, a few N incoherent measurements are enough, *so* $N = P\ (Flogm)$.

CS addresses the challenge of acquiring prior knowledge of the precise correlation of the sensed data, which may not always be available in many *WSN* applications. This is because this technique exploits the compressibility of sensed data without prior knowledge of the data. The theory behind *CS* states that for a signal that can be compressed, the original signal can be gathered and re-constructed with lesser samples than required by the Nyquist rate [44]. The Nyquist theorem states that to perfectly recover the source signal, the sampling frequency should at least be twice the highest frequency of the source signal. With *CS*, a reduction in sampling frequency leads to a decrease in energy consumption. The sampling

TABLE 1. A summary of comparison for local data compression techniques.

FEATURE	**LEC** [4].	**TMT** [33], [49]	**ALDC** [2]	**FELACS** [3]
a. Compression ratio.	@45 – 75%	@40%	52.8 – 73.9%	41 – 73.8%
2. Energy saved: minimized transmission	@53%	@40%	NA	NA
3. Processing Complexity	12 instructions per saved bit	4 additions, 2 integer multiplications, 2 shifts, 2 comparisons.	simple	Lightweight.
4. Net Energy Saving.	@32%	@36%	NA	More than 55%
5. Lossless. / Lossy	Lossless	Lossless	Lossless.	Lossless
6. Limitations	Does not exploit spatial correlations.			
7. Data Types: single/multiple	Multiple	Single	Multiple.	Multiple

The compression ratios for *ALDC* and *FELACS* scheme are much higher than that of *LEC* and *TMT*. *FELACS* is lightweight and saved more than 55% energy due to its fast feature in compressing data. *TMT* demonstrated the lowest compression ratio and can only be applied to a single data type for every node.

frequency relies on the signal characteristics and not on the bandwidth [18]. Suitable conditions for *CS* are that the signal should be compressible, as in *WSN* data, where source nodes are constrained and allow simple coding, but complex decoding is realized at the sink, which is not limited in energy supply and processing capability [45]. A naturally sparse signal can be reconstructed by applying a few linear measurements. An l1-norm convex optimization problem is solved [18].

An efficient *CS* scheme provides a tradeoff between the measurement cost and the quality of data recovery. The main challenge in *CS* is the computational complexity during the reconstruction phase, particularly for real-time deployments for large scale *WSNs*. It must also consider the network size and compression during the design of the network.

Authors in [46] introduced compressed sensing theory for aggregation of data in a multi-hop wireless sensor network. The data gathering challenge faced by *WSNs* in which routing and compression were simultaneously employed to transfer random data projections were addressed [18], [45]. The latter demonstrated the concept with simulations of artificial sample signals. The former used both artificial and real-world data to quantify the compressed sensing performance. Even though results from both authors indicated that compressed sensing performed better than routing schemes, there remained a challenge of discovering a matrix that would make the signal sparse. *DCT,* Harr wavelet transformation, and their proposed transformations were experimented with but failed to sparsify the signal [6] To further reduce energy consumption, authors [46] proposed an efficient data gathering framework that uses sparse projections that are spatially localized and collected energy of transform coefficients more evenly distributed. The theory of adaptive compressed sensing was introduced by [47], [48] to collect information from the network efficiently. The approaches faced the challenge of having to maximize the volume of data gathered for every energy expenditure. Using heuristic schemes, performance simulations and field tests resolved this problem by providing a more precise approximation of the temporal, spatial fields for the desired energy expenditure.

Table 1 and Table 2 summarize reported performances of reviewed data compression (Local data compression and Distributed data compression) algorithms for comparison. The criteria or measures used for comparison are explained in section 4.

IV. PERFORMANCE ANALYSIS OF DATA COMPRESSION ALGORITHMS

Local data compression algorithms are robust and universal. They can be integrated with distributed data compression algorithms to exploit spatial and temporal correlations inherent in sensed data.

A. PERFORMANCE MEASURES

The following measures are applied to analyze the performance of different data compression algorithms:

1) COMPRESSION RATIO

Using values reported from previous reviewed research work, a summary of their compression ratios is presented for comparison.

a: ENERGY SAVED DUE TO MINIMIZED TRANSMISSION

This is a measure of the amount at which reduction of transmission is affected by compression ratios of every algorithm. Energy saving because of compression ratio is presented and compared.

b: ENERGY CONSUMED DURING DATA COMPRESSION

This analyzes the power used when compressing data. The complexity of compression algorithms is assessed within

TABLE 2. A summary of comparison for distributed data compression techniques.

FEATURE	DSC				DSM		DTC: IMAGE BASED			CS
	EEADSC [50]	[51]	**DSC-multi-rate** [52]	***OQET/ OQPT*** [53][19]	**DRHMM/VFBSN** [54][55]	**DNKB** [42][43]	**KLT** [14]	**DWT-lifting** [15]	**DWT-Harr** [17]	[16], [18], [47], [48]
Compression ratio.	~88%	NA	Dependent on packet error	Optimal	NA	NA	NA	Depends on SNR	Depends on SNR	Dependent on data sparsity.
Energy saved: minimized transmission.	~88%	NA	Need to know correlation among nodes precisely	YES	YES	YES	YES	YES	YES	YES
Processing Complexity	~22%	Low cost	Low cost	$Q(L_b^2)$	Q(N)/ low cost.	Q(N)	Low cost	Low cost	Low cost	Low cost
Net Energy Saving	~66%	50.7%	YES	YES	YES	YES	YES	YES	YES	YES
Lossless / Lossy	Lossless	Lossless	Lossless	Lossy	Lossy	Lossy	Lossless	Lossless	Lossless	Lossy
Limitations	Limited to star topologies	Need to precisely know correlation among nodes.	Need to know correlation among nodes precisely	Limited to specific application	Limited to specific application	Limited to specific application	Limited to specific topologies and needs extra inter-sensor data exchange	Limited to specific topologies and needs extra inter-sensor data exchange	Limited to specific topologies and needs extra inter-sensor data exchange	Suitable transformation is needed to improve sparsity for real world data.
Data Types: single/multiple	Single	Single	Single	Single	Single	Single	Single	Single	Single	[18] experimented on suitable multiple data types.

its class. The number of basic coding operations in every algorithm can be counted to compare the processing complexity. The energy expended in executing these instructions can then be presented.

c: NET ENERGY SAVED

The difference between energy saved due to minimized transmissions and energy used for running the algorithm. It is a vital determinant for suitable algorithms for different applications in WSNs.

d: SUITABLE CLASSES OF DATA COMPRESSION - LOSSLESS/LOSSY

It analyzes and informs if an algorithm is a proper design for its equivalent data [6].

e: DATA TYPES (SINGLE/MULTIPLE)

This criterion tells if an algorithm can be used for a single data type for each node or multiple data types.

Communication between sensor nodes is not needed for *DSC* algorithms. *Energy-Efficient Adaptive Distributed Source Coding (EEADSC)* outperforms the other *DSC* algorithms reported here regarding compression ratio and net energy saving. It is however limited to star topologies.

Compression ratios for *DSM* schemes, *Optimizing Quantization-based Estimation Target (OQET)* and *Optimizing quantization-based power target (OQPT)* are reliant on the quantized data bit length L_b, which is dictated by an optimization solution. For these algorithms, a processing complexity of $Q(Lb^2)$ is needed by sensor nodes to discover an optimal bit length. This may be too high for sensor nodes for higher values of L_b. For *Data Representation-based Hidden Markov Model (DRHMM), Variational Filtering in Binary Sensor Network (VFBSN)* and *Distributed Non-Parametric Kernel-Based scheme (DNKB),* a processing complexity of only *Q(N),* which is lower, is needed. Although these *DSM* algorithms can minimize communication data in sensor networks, they are lossy and cannot be applied to some WSN applications that require high accuracy in sensed data.

Distributed Transform Coding (DTC), is common for video and image compression. Unlike *DSM*, it transforms data into coefficients, which are represented by suitable codes. *Distributed Karhunen–Loeve transform (KLT)* processes data at the sink node that is not energy constrained and saves energy. The compression ratio and amount of energy saving in *KLT*, *Distributed Wavelet Transform-based lifting (DWT-lifting)* and *Distributed Wavelet transform-based Harr (DWT-Harr)* depend on the signal to noise ratio (*SNR*) in the network. Still, *DWT*-Harr outperforms all the other algorithms in compressing data.

CS data compression algorithms do not need any prior knowledge on the data to be compressed. Algorithms that used real-world data sets [16], [18] for their experiments demonstrated poor results than those that used ideal data, but error recovery for given energy consumption for these schemes was lower than 0.3 and acceptable. Where multiple data types were used [16], better performance was evidenced.

From the comparison made on *Local data* compression algorithms, a lossless local data compression algorithms (*ALDC)* was selected for performance evaluation in this work.

B. ANALYSIS OF AN ADAPTIVE LOSSLESS DATA COMPRESSION ALGORITHM (ALDC) [2]

ALDC can acclimatize to fluctuations in the properties of the source data and compress data before it can be sent. Blocks of sampled data are compressed at a time using two coding options that use lossless entropy compression code options named *2 Huffman-Table ALEC (Adaptive Lossless Entropy Compression)* and *3 Huffman Table ALEC* [2].

As illustrated in Table 3, the Huffman tables in these code options were designed using practical wireless sensor node datasets with different states of correlation. The two code options have associated algorithms to encode blocks of sampled data. *ALDC* uses a simple and efficient predictive coding method to obtain sampled data temporal correlations and compress environmental data such as relative humidity, temperature, and seismic data. For example, for a given block of temperature data samples $xi = \{8202,8202,8202,8201,8202,8202,8202,8208\}$, each sample can be represented by 14 bits, which is the resolution of the analogue to digital converter (*ADC*) of the encoder. To send all the 8 samples translates to $14 \times 8 = 112$ bits that will need to be sent to the encoder. The goal of the *ALDC* algorithm is to reduce the number of bits that are needed to represent each sample. Instead of sending the sample readings from the sensor nodes, the difference between consecutive sampled data (residues) is sent, reducing the dynamic range of source symbols. Huffman coding is then used to encode the samples and determine how the samples are going to be sent. The first sample of the data samples is calculated

$$\text{as } x_0 = 2^{DR-1}, \tag{3}$$

where *DR* is the dynamic range of the source data, which is the number of bits needed to encode the data samples. The predicted sample is equal to the last measured sample and is calculated as

$$\hat{x} = x_{i-1} \tag{4}$$

The residue res can be calculated from this sample by subtracting the predicted sample from the current one.

$$res = x_i - x_{i-1}. \tag{5}$$

This residue is the input to the entropy encoder that uses two *ALEC* code options, the *Brute Force* and *Decision regions* coding schemes. The *Brute Force* coding scheme encodes bitstreams for both code options, demanding more memory space and hardware, showing more complex data processing. However, the Decision regions coding scheme uses only one

TABLE 3. (a) ALDC coding tables: Huffman table A. (b) ALDC coding tables: Huffman table B. (c) ALDC coding tables: Huffman table C.

(a)

b_i	h_i	d_i
0	00	0
1	01	$-1, +1$
2	11	$-3, -2, +2, +3$
3	101	$-7, \ldots, -4, +4, \ldots, +7$
4	1001	$-15, \ldots, -8, +8, \ldots, +15$
5	10001	$-31, \ldots, -16, +16, \ldots, +31$
6	100001	$-63, \ldots, -32, +32, \ldots, +63$
7	1000001	$-127, \ldots, -64, +64, \ldots, +127$
8	10000001	$-255, \ldots, -128, +128, \ldots, +255$
9	1000000000	$-511, \ldots, -256, +256, \ldots, +511$
10	10000000010	$-1023, \ldots, -512, +512, \ldots, +1023$
11	10000000011	$-2047, \ldots, -1024, +1024, \ldots, +2047$
12	10000000100	$-4095, \ldots, -2048, +2048, \ldots, +4095$
13	10000000101	$-8191, \ldots, -4096, +4096, \ldots, +8191$
14	10000000110	$-16383, \ldots, -8192, +8192, \ldots, +16383$

(b)

b_i	h_i	d_i
0	1101111	0
1	11010	$-1, +1$
2	1100	$-3, -2, +2, +3$
3	011	$-7, \ldots, -4, +4, \ldots, +7$
4	111	$-15, \ldots, -8, +8, \ldots, +15$
5	10	$-31, \ldots, -16, +16, \ldots, +31$
6	00	$-63, \ldots, -32, +32, \ldots, +63$
7	010	$-127, \ldots, -64, +64, \ldots, +127$
8	110110	$-255, \ldots, -128, +128, \ldots, +255$
9	110111011	$-511, \ldots, -256, +256, \ldots, +511$
10	110111001	$-1023, \ldots, -512, +512, \ldots, +1023$
11	1101110101	$-2047, \ldots, -1024, +1024, \ldots, +2047$
12	1101110100	$-4095, \ldots, -2048, +2048, \ldots, +4095$
13	1101110000	$-8191, \ldots, -4096, +4096, \ldots, +8191$
14	11011100011	$-16383, \ldots, -8192, +8192, \ldots, +16383$

(c)

b_i	h_i	d_i
0	1001	0
1	101	$-1, +1$
2	00	$-3, -2, +2, +3$
3	01	$-7, \ldots, -4, +4, \ldots, +7$
4	11	$-15, \ldots, -8, +8, \ldots, +15$
5	10001	$-31, \ldots, -16, +16, \ldots, +31$
6	100001	$-63, \ldots, -32, +32, \ldots, +63$
7	1000001	$-127, \ldots, -64, +64, \ldots, +127$
8	10000001	$-255, \ldots, -128, +128, \ldots, +255$
9	1000000000	$-511, \ldots, -256, +256, \ldots, +511$
10	10000000010	$-1023, \ldots, -512, +512, \ldots, +1023$
11	10000000011	$-2047, \ldots, -1024, +1024, \ldots, +2047$
12	10000000100	$-4095, \ldots, -2048, +2048, \ldots, +4095$
13	10000000101	$-8191, \ldots, -4096, +4096, \ldots, +8191$
14	10000000110	$-16383, \ldots, -8192, +8192, \ldots, +16383$

code option, requiring less hardware and processing power. This option utilizes a table of decision regions that relies on the length of the original sequence of sampled data.

Using the Decisions region approach, residues are calculated as {10,0,0,-1,1,0,0,6}. The *MATLAB Pseudocode I* below is used to writing a function called '*Residual*', which calculates the differences between consecutive samples.

Pseudocode 1 Residual (*data, DR, xo, residue*)

```
//data is the original source data.
//DR is the dynamic range of source data.
Compute the first residue.
//xo is the first residue
Set residue(i) to source(i) − xo.
//Compute the residues for every length of source data.
RETURN residue.
```

The sum of the absolute value of the residues is

$$F = \sum_{i=1}^{K} |res| = 10 + 0 + 0 + 1 + 1 + 0 + 0 + 6 = 18 \quad (6)$$

The length of the sequence of residues is 8 samples. For the decision regions, the first boundary is $3K = 3 \times 8 = 24$, where K denotes the number of residue samples. The second boundary is $12K = 12 \times 8 = 96$.

Since the sum of the residues ($F = 18$), $F < 24$ and lies in the first boundary region, the *ALEC* code option that uses two Huffman tables A and B called 2-Huffman is used to encode the residues and a code identifier *ID*, '0' is generated for it. A code option identifier '1' is generated for any sum that falls outside this region and selects the 3-Huffman Table *ALEC* (5). Pseudocode 2 creates a function called '*adaptive*,' which calculates the residues' length, the sum of absolute values, and generates a code option identifier.

Pseudocode 2 Adaptive(*F,K, ID*)

```
//Adaptive is the function that determines the boundary for
the decision regions.
//Compute F, the total of the absolute value of residues.
//K denotes the residue length.
Call Residual() and calculate K.
//Determine the F region.
IF $F < 3K$
//Set code ID for using two Huffman Tables A, B
ID → '0'.
ELSE
IF $3K < F <= 12K$
//Set ID for using three Huffman tables A, B and C
ID → '1'.
ENDIF
RETURN ID.
```

The *'My_encoder'* pseudocode is used to encode residues' samples. Pseudocode '*Huff_2*' is used to generate the MATLAB CODE that uses 2 Huffman Table *ALEC* while '*Huff_3*' generates the code that uses 3 Huffman Table ALEC to encode. The encoded bitstreams are concatenated with the previously generated code option *IDs*. The sizes of the bitstreams are compared to select the least compressed.

Pseudocode 3 My_encoder (*residue_value*, HUFF, *ci*) *[2]*

```
// HUFF is the Huffman table to be used for encoding.
// bi equals residue_value group identifier and also the
number of least order bits that are required to represent that
residue_value in binary.
// ci is the bitstream that has been encoded.
//A → B means set A to B
// hi is the codeword from the Huffman table used to encode
the residue_value.
// li is the codeword that represents the position of the
residue_value in its group.
// ∗ illustrates concatenation
// (Index)| bi = index over bi bits as a binary number
// compute residue_value category
IF residue_value = 0 THEN
bi → 0
ELSE
bi → log2 (|residue_value|)_
ENDIF
// get hi from HUFF
hi → HUFF [bi]
// build ci
IF bi = 0 THEN
// no need to calculate li here
ci → hi
ELSE
// compute li
li → (Index)| bi
// compute ci
ci → hi ∗ li
ENDIF
RETURN ci
```

TABLE 4. Output bitstream of ALDC.

Index Position *bi*	Category/Group *bi*	Binary code representation *hi*	Residue Sample *di*
4	1001	1010	10
0	00	-	0
0	00	-	0
1	01	0	-1
1	01	1	1
0	00	-	0
0	00	-	0
3	101	110	6

C. RESULTS AND ANALYSIS OF THE ALDC ALGORITHM

The encoded bitstream is an output "0 0 1001 1010 00 00 01 0 01 1 00 00 101 110", which has a length of 30 bits for the given block of 8 data samples. The first red '0' is a code identifier *ID* that informs the decoder that a *ALEC* code option that uses two Huffman Tables was used. The second green '0' is a table identifier *ID* that informs the decoder that Huffman coding Table A in Table 3 that was used by the *ALEC* code option that uses two Huffman tables to encode the 8 samples. Table 4 below explains the rest of the numbers in the output bitstream.

Pseudocode 4 Huff_2 *(residue_vector, table1, table2*, code)

```
// My_encoder() is the My_encoder function
// code is the encoded bitstream of N residue_vector.
//A → B means set A to B

// ∗ denotes concatenatio
// The block of K residue_vector is encoded using the first
Huffman Table of the 2-Huffman Table ALEC Coder
CALL My_encodeer() to encode block of K residue_vector
using Table A to RETURN cil.
ciA → cil
// append encoded bitstream ciA to code1
code1 → code1*CiA
// encode the same block of K residue_vector using the second Huffman Table of the 2-Huffman Table ALEC Coder
CALL My_encoder()to encode a block of K residue_vector
using Table B to Return ci2
ciB → ci2
// append encoded bitstream ciB to code2
code2 → code2*CiB
// compute the length of the encoded bitstream code1
// compute the length of the encoded bitstream code2
// compare length of code1 and length of code2 and select
the encoded bitstream with the least compressed size
IF length code1 <= length code2 THEN
// generate ID of Table A
ID → "0"
// concatenate code1 to ID
code → ID∗ code1
ELSE
// generate ID of Table B
ID → "1"
// concatenate code2 to ID
code → ID∗ code2
ENDIF
RETURN code
```

TABLE 5. Encoded output for different block sizes.

BLOC K SIZE	OUTPUT +BITSTREAM	NO OF BIT S	BITS SAVE D	ENER GY SAVIN G
1	'01111101000000000100011000000 0101110'	36	76	67.85%
2	'00100110100000000100011000001 01110'	33	79	70.5%
3	'00100110100000000100110000010 1110'	32	80	71.4%
4	'00100110100000001000110000101 110'	31	81	72.3%
5	'00100110100000001001100000101 110'	31	81	72.3%
6	'00100110100000001001100000101 110'	31	81	72.3%
7	'00100110100000001001100000101 110'	31	81	72.3%
8	'00100110100000001001100001011 10'	30	82	73.2%

When $di = 0$, $bi = 0$ and the binary representation is not required. The encoding procedure was repeated for different block sizes of the length of these samples, which can be 1,2,3,4,5,6,7 and 8. The results of the output bitstream are as tabulated on Table 5.

Pseudocode 5 Huff_3 (*residue_vector*, *table1,table2,table3*, lucia_code)

```
// My_encoder() is the encoding function
// lucia_code represents the encoded bitstream of K
residue_vector.
//A → B means set A to B

//∗ is the concatenation operatio
// block of K residue_vectors is encoded using TABLE A
CALL My_encoder() to encode a block of K
residue_vector using Table A to produce cil
ciA → cil
// append encoded bitstream ciA to code_1
code_1 → code_1*CiA
// The same block of K residue_vector is encoded using
TABLE B
CALL to encode a block of K residue_vector using Table B
to produce ci2
ciB → ci2
// append encoded bitstream ciB to code_2
code_2 → code_2*CiB
// The same block of K residue_vector is encoded using
TABLE C
CALLMy_ encoder() to encode a block of K
residue_vector using Table C to produce ci3
ciC → ci3
// append encoded bitstream ciC to code_3
code_3 → code_3*CiC
// compute the length of the encoded bitstream code_1
// compute the length of the encoded bitstream code_2
// compute the length of the encoded bitstream code_3
// compare length of code_1, code_2, code_3 and select the
encoded bitstream with the least compressed size
IF length code_1 <= min(length code_2 && length
code_3) THEN
// ID to show that Table A was used
ID → "10
//ciA is appended to ID
lucia_code → ID ∗ code_1
ELSEIF length code_2 <= min(length code_1 && length
code_3) THEN
// ID for to show that Table B was used
ID → "11
// concatenate ciB to ID
lucia_code → ID ∗ code_2
ELSEIF length code_3 <= min(length code_2 && length
code_1) THEN
// ID to tell the encoder that Table C was used
ID → "0"
// concatenate ciC to ID
lucia_code → ID∗ code_3
ENDIF
RETURN lucia_code
```

The results show a reduction of number of bits that are to be sent to the decoder. This translates to an energy saving of the sensor nodes that can be calculated as:

$$\text{Energy Saving} = \frac{\textit{Energy saved through compression } E_{Comp}}{\textit{Energy consumed without compression } E_{Uncomp}} \times 100. \tag{7}$$

If it is assumed that the energy consumed by processing is less significant than the energy consumed due to transmission, the sensor node energy saving can be approximated to the energy saving of the transmission module, which is

$$\text{Energy Saving} = \frac{\textit{Average number of bits saved per sample } N_b}{\textit{Number of bits per sample of original data } N} \times 100. \tag{8}$$

Therefore, the energy saving for every block size is calculated, and the results are shown in *Table 4*. For a block size of 8 samples,

Energy Saving $= \frac{82}{112} \times 100 = 73.2\%$. This can be expressed as a compression ratio (*CR*)

$$\begin{aligned} CR &= (1 - \frac{\textit{Compressed data}}{\textit{raw data}}) \times 100\% \\ &= (1 - \frac{30}{112}) \times 100 = 73.2\%. \end{aligned} \tag{9}$$

The graph illustrating the compression ratio for the different block sizes is as shown by Fig. 10.

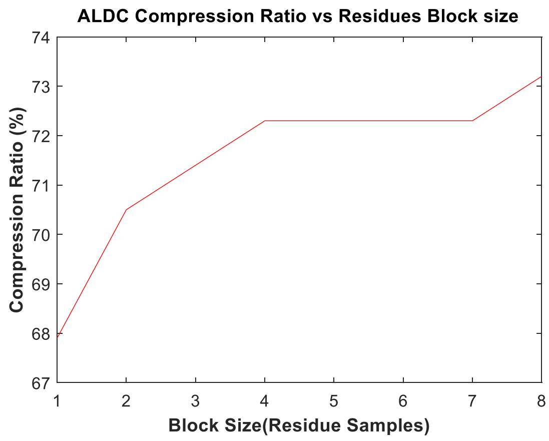


FIGURE 10. Compression ratio vs residues block sizes for the ALDC algorithm.

The results indicate that more energy is saved when the data samples are divided into bigger blocks. This confirms that the compression performance of the *ALDC* algorithm increases with the increase in the block size. It is also noted that more energy has been saved by encoding residues, rather than encoding the original individual readings from the sensor nodes. Fig 11 confirms that as the number of bits decreases, the energy-saving increases. Fig 12 demonstrates that as the number of bits saved from the original data stream increase, the energy-saving also increases, showing better compression performance. This analysis, however, was based on the test data provided, not from real-world data sets used in comparison with test data results.

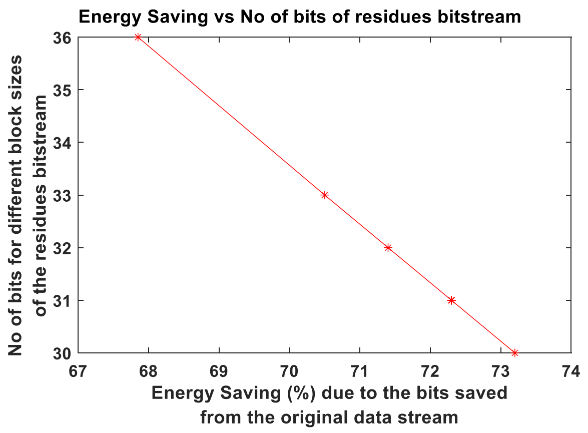


FIGURE 11. Number of bits for residue samples vs energy saving for ALDC algorithm.

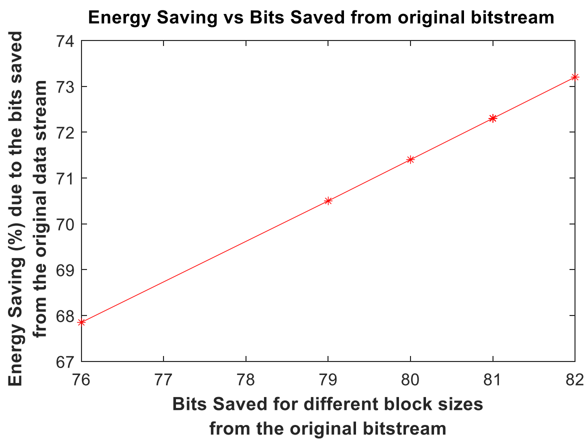


FIGURE 12. Number of bits saved from the original ALDC bitsream vs energy saving of bits saved from the original bitstream of the ALDC algorithm.

A measure of the average number of binary symbols needed to encode a source's output is a quantity called 'Entropy'. This is the average self-information that is associated with a random experiment. It is usually estimated based on the assumptions made about the structure of the sequence of data samples. An efficient lossless compression technique encodes the output of a source using an average number of bits that is equal to the entropy of that source. Considering our example of incoming temperature data samples *xi = {8202,8202,8202,8201,8202,8202,8202,8208},* the information entropy H can be calculated as

$$H = -\sum_{i=1}^{N} P_{xi}.log_2 P_{xi}, \tag{10}$$

where P_{xi} is the probability mass function of *xi,* and N is the number of possible values of the 14 bit analogue to digital converter *xi.*

TABLE 6. Huffman coding using tables A, B, and C.

bi	TABLE	*hi*	residues
0	1	00	0
1	1	01	-1,+1
2	1	11	-3,-2,+2,+3
3	3	01	-7,…,4,+4,…,+7
4	3	11	-15,..,-8,+8,..,+15
5	2	10	-31,…,-16,+16,…,+31
6	2	00	-63,…,-32+32,…,+63
7	2	010	-127,…,-64,+64,…,+127
8	2	110110	-255,…,-128,+128,…,+255
9	2	110111011	-511,…,-256,+256,…,+511
10	2	110111001	-1023,…,-512,+512,…,+1023
11	2	1101110101	-2047,…,-1024,+1024,…+2047
12	2	1101110100	-4095,…-2048,+2048,…+4095
13	2	1101110000	-8191,…,-4096,+4096,…,+8191
14	2	11011100011	-16383,…,-8192,+8192,…,+16383

The probability

$$P_{(8202)} = 6/8. \quad H_{(8202)} = -6/8 \times \log2(6/8) = 0.3113.$$
$$P_{(8201)} = 1/8. \quad H_{(8201)} = -1/8 \times \log2(1/8) = 0.375.$$
$$P_{(8208)} = 1/8. \quad H_{(8208)} = -1/8 \times \log2(1/8) = 0.375.$$

$H = 3.113 + 0.375 + 0.375 = 1.06$ bits/sample. This means that this sequence could best be encoded at 1.06 bits per sample. The average length is $0.75 \times 14 + 0.125 \times 14 + 0.75 \times 14 =$ 14bits/symbol. The information entropy of the residues H_{res} is calculated as

$$H_{res} = -\sum_{i=1}^{N} P_{res}.log_2 P_{res}. \tag{11}$$

The probability

$$P_{(10)} = 1/8 \quad H_{(10)} = -1/8 \times \log2(1/8) = 0.375.$$
$$P_{(0)} = 4/8. \quad H_{(0)} = -4/8 \times \log2(4/8) = 0.5.$$
$$P_{(-1)} = 1/8. \quad H_{(-1)} = -1/8 \times \log2(1/8) = 0.375.$$
$$P_{(1)} = 1/8. \quad H_{(1)} = -1/8 \times \log2(1/8) = 0.375.$$
$$P_{(6)} = 1/8. \quad H_{(6)} = -18 \times \log2(1/8) = 0.375.$$
$$H_{res} = 0.375 \times 4 + 0.5 = 2\text{bits/symbol}.$$

The average length of the code l is $0.125 \times 4 + 0.5 \times 0 + 0.125 \times 1 + 0.125 \times 1 + 0.125 \times 3 = 1.125$. The difference between entropy and the average length (2-1.125= 0.875) is the measure of the efficiency of the code, which is the redundancy. The decoder needs to know the process that generated the sequence of residues from the original sequence. This process relies on the model of the sequence, which is the assumptions made regarding the sequence structure.

V. THE PROPOSED ALGORITHM

This work introduces an algorithm that modifies the original *ALDC* algorithm and save more energy than the original algorithm. Instead of encoding data samples using the two code options, the algorithm uses all the three tables interchangeably as shown in TABLE 6. After analysis of the performance of the original *ALDC* algorithm, it is realized that all three *ALDC* Huffman coding tables show regions of *bi*, where less bits of data were needed to encode the samples of residues.. The new approach realizes that compression performance can further be improved by interchangeably implementing the three Huffman Coding tables A, B and C, utilizing their *bi* regions where shorter codewords were used to encode residue samples.

TABLE 7. Binary code representations for residue samples showing selected Huffman tables.

Residue sample *di*	Index Position *bi*	Binary Code rep *hi*	TABLE
10	4	11	3
0	0	-	1
0	0	-	1
-1	1	01	1
1	1	01	1
0	0	-	0
0	0	-	0
6	3	01	2

Table 7 draws relevant information from table 6 and illustrates the residue samples, their index position bi on suitably selected tables, and associated binary code representation. It can be noted from Table 6 that to encode data samples '10, 0, 0 -1, 1, 0, 0, 6', only two bits are needed for each symbol. As in the ALDC algorithm, when di = 0, bi = 0 and the binary representation hi is not required.

The average length of this code is 0.125*2+0.5*2+0.125*2+0.125*2+0.125*2, which is 2bits/symbol. The average length is equal to the entropy of the data samples, confirming that this lossless data compression technique is efficient.

Modification to the *ALDC* algorithm is achieved by implementing the '*Huff_4*' algorithm, which uses '*My_encoder*' pseudocode to perform the encoding process. The results of the output bitstream produced from this improvement are as shown in Table 8.

Pseudocode 6 *Huff_4* (residue_vector, table1, table2, table3, code_1)

```
//My_encoder() is the encode function
//code_1 is the encoded bitstream of N residue_vector.
//* illustrates concatenation
//encode block of N residue_vector using the three Huffman Tables.
CALL My_encoder() with block of N residue_vector and Tables A, B and C RETURNING ci1.
ciA → ci1
// append encoded bitstream ciA to code_1
code_1 → code_1*ciA
RETURN code_1
```

TABLE 8. Results of the modified encoding algorithm.

BLOCK SIZE	OUTPUT BITSREAM	NO OF BITS	BITS SAVED FROM ALDC	ENERGY SAVING FROM RAW DATA (%)	ENERGY SAVING FROM ALDC (%)
1	'0111010000001 0011000001110'	26	10	76.78	27.8
2	'0111010000001 0011000001110'	26	7	76.78	21.2
3	'0111010000001 0011000001110'	26	6	76.78	18.75
4	'0111010000001 0011000001110'	26	5	76.78	16.13
5	'0111010000001 0011000001110'	26	5	76.78	16.13
6	'0111010000001 0011000001110'	26	5	76.78	16.13
7	'0111010000001 0011000001110'	26	5	76.78	16.13
8	'0111010000001 0011000001110'	26	4	76.78	13.33

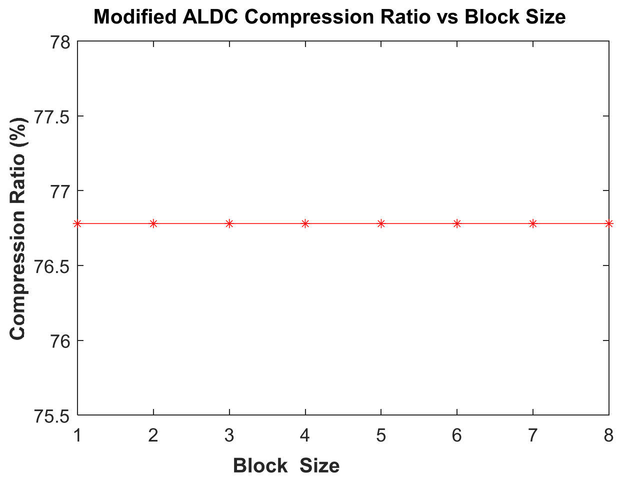


FIGURE 13. Compression ratio vs residues block sizes for the modified ALDC algorithm.

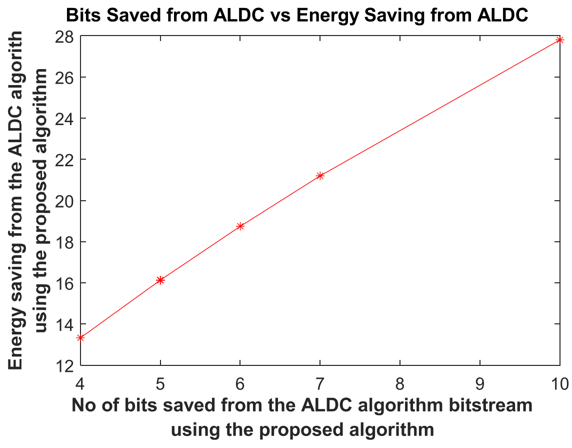


FIGURE 14. Number of bits saved from the ALDC output bitstream of the modified ALDC algorithm.

Table 8 shows that the proposed approach resulted in a 26-bit output bitstream for all block sizes of residue samples compared to the 30 - 36 bits output bitstream of the *ALDC*.

Pseudocode 7 *My_Encoder (residue_value, table1, table2, table3, ci)*

```
// tables1, table2 and table3 are the encoding Huffman tables.
// bi is the group number of residue_value and also the number of least order bits that are required to encode the residue_value.
// ci is the output bitstream that has been encoded.
// hi is the Huffman code that represents the residue_value group.
// li is the variable-length integer code that encodes the index position of the residue_value in its group.
// * illustrates concatenation
// (Index)| bi = index over bi bits as a binary number over bi bits
//Compute residue_value
IF $residue\_value = 0$ THEN
$bi \to 0$
ELSEIF $|residue\_value| = 1$
$bi \to 1$
ELSE
$bi \to$ _log2 (| residue_value|)_
ENDIF
// extract hi the variable length Huffman code from TABLES
IF $bi = 0$ || $bi = 1$ || $bi = 2$ THEN
$hi \to$ table1 [bi]
ELSEIF $bi = 3$ || $bi = 4$ THEN
$hi \to$ table3 [bi]
ELSE
$hi \to$ table2 [bi]
ENDIF
// build ci
IF $bi = 0$ THEN
// li is not needed
$ci \to hi$
ELSEIF $residue\_value < 0$ THEN
$li \to$ Binary
$ci \to hi*li$
ELSE
// build li
$li \to$ (Index)| bi
// build ci
$ci \to hi * li$
ENDIF
RETURN ci
```

This is energy saving of 4 to 10 bits, which translates to a total energy saving of 76.78%. The results are as plotted in Fig 13.

As the number of bits saved by modifying the ALDC algorithm increases, the energy-saving also increases, showing improvement in data compression performance as illustrated by Fig14. The codes used for this paper can be found in [56].

VI. CONCLUSION

This research classified data compression techniques into local data compression, data aggregation and distributed data compression techniques. A survey and comparison of existing data compression algorithms is presented, and an *ALDC* algorithm was analyzed by simulation in *MATLAB*. The simulation results suggest that the Huffman tables used to encode data residues can significantly compress the data before it can be applied to the encoder. The authors' proposed an approach

that uses all the three Huffman coding tables interchangeably to encode residues samples of data. The new approach further reduced the *ALDC* output bitstream ranging from 30 to 36 bits to a significant 26 bits. This is an improved energy saving of 76.78%, demonstrating a significant improvement of data compression performance to the algorithm under analysis and review. This has satisfied this work's aim of discovering strategies that can be used to reduce the amount of data before it can be transmitted. In the future the proposed method can be extended by generating own Huffman tables to further reduce energy consumption. The method can be applied to compress real world data sets. In Addition, the proposed algorithm is to be used in conjuction with routing algorithm (Termite-hill) to achieve a new compressed routing algorithm for wireless sensor networks.

REFERENCES

[1] L. K. Ketshabetswe, A. M. Zungeru, M. Mangwala, J. M. Chuma, and B. Sigweni, "Communication protocols for wireless sensor networks: A survey and comparison," *Heliyon*, vol. 5, no. 5, May 2019, Art. no. e01591, doi: 10.1016/j.heliyon.2019.e01591.

[2] J. G. Kolo, S. A. Shanmugam, D. W. G. Lim, L.-M. Ang, and K. P. Seng, "An adaptive lossless data compression scheme for wireless sensor networks," *J. Sensors*, vol. 2012, pp. 1–20, Jul. 2012, doi: 10.1155/2012/539638.

[3] J. G. Kolo, S. A. Shanmugam, D. W. G. Lim, and L.-M. Ang, "Fast and efficient lossless adaptive compression scheme for wireless sensor networks," *Comput. Electr. Eng.*, vol. 41, pp. 275–287, Jan. 2015, doi: 10.1016/j.compeleceng.2014.06.008.

[4] F. Marcelloni and M. Vecchio, "An efficient lossless compression algorithm for tiny nodes of monitoring wireless sensor networks," *Comput. J.*, vol. 52, no. 8, pp. 969–987, 2009, doi: 10.1093/comjnl/bxp035.

[5] N. Kimura and S. Latifi, "A survey on data compression in wireless sensor networks," in *Proc. Int. Conf. Inf. Technol., Coding Comput. (ITCC)*, vol. 2, Apr. 2005, pp. 8–13, doi: 10.1109/itcc.2005.43.

[6] T. Srisooksai, K. Keamarungsi, P. Lamsrichan, and K. Araki, "Practical data compression in wireless sensor networks: A survey," *J. Netw. Comput. Appl.*, vol. 35, no. 1, pp. 37–59, Jan. 2012, doi: 10.1016/j.jnca.2011.03.001.

[7] K. Sayood, *Introduction to Data Compression* (Morgan Kaufmann Series in Multimedia Information and Systems), 3rd ed. Lincoln, NE, USA: Univ. Nebraska, 2006.

[8] M. A. Razzaque, C. Bleakley, and S. Dobson, "Compression in wireless sensor networks: A survey and comparative evaluation," *ACM Trans. Sensor Netw.*, vol. 10, no. 1, pp. 1–44, Nov. 2013, doi: 10.1145/2528948.

[9] Y. C. Wang, "Data compression techniques in wireless sensor networks," *Pervas. Comput.*, vol. 6, no. 1, pp. 61–86, 2012.

[10] M. Ding, X. Cheng, and G. Xue, "Aggregation tree construction in sensor networks," in *Proc. IEEE 58th Veh. Technol. Conf. VTC-Fall*, Oct. 2003, pp. 2168–2172, doi: 10.1109/VETECF.2003.1285913.

[11] R. Rajagopalan and P. K. Varshney, "Data-aggregation techniques in sensor networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 8, no. 4, pp. 48–63, Oct. 2006, doi: 10.1109/COMST.2006.283821.

[12] S. Ozdemir and Y. Xiao, "Secure data aggregation in wireless sensor networks: A comprehensive overview," *Comput. Netw.*, vol. 53, no. 12, pp. 2022–2037, Aug. 2009, doi: 10.1016/j.comnet.2009.02.023.

[13] H. Karl and A. Willig, "Protocols and architectures for wireless sensor networks," *Protoc. Archit. Wireless Sens. Netw.*, pp. 1–497, Apr. 2006, doi: 10.1002/0470095121.

[14] A. Amar, A. Leshem, and M. Gastpar, "Recursive implementation of the distributed Karhunen-Loève transform," *IEEE Trans. Signal Process.*, vol. 58, no. 10, pp. 5320–5330, Oct. 2010, doi: 10.1109/TSP.2010.2056922.

[15] A. Ciancio and A. Ortega, "A distributed wavelet compression algorithm for wireless multihop sensor networks using lifting," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process., (ICASSP)*, Mar. 2005, pp. 825–828, doi: 10.1109/ICASSP.2005.1416136.

[16] C. T. Chou, R. Rana, and W. Hu, "Energy efficient information collection in wireless sensor networks using adaptive compressive sensing," in *Proc. IEEE 34th Conf. Local Comput. Netw.*, Oct. 2009, pp. 443–450, doi: 10.1109/LCN.2009.5355162.

[17] G. Shen and A. Ortega, "Transform-based distributed data gathering," *IEEE Trans. Signal Process.*, vol. 58, no. 7, pp. 3802–3815, Jul. 2010, doi: 10.1109/TSP.2010.2047640.

[18] G. Quer, R. Masiero, D. Munaretto, M. Rossi, J. Widmer, and M. Zorzi, "On the interplay between routing and signal representation for compressive sensing in wireless sensor networks," *Inf. Theory Appl. Work. ITA*, vol. 2009, pp. 206–215, Mar. 2009, doi: 10.1109/ITA.2009.5044947.

[19] J. Li and G. Alregib, "Distributed estimation in energy-constrained wireless sensor networks," *IEEE Trans. Signal Process.*, vol. 57, no. 10, pp. 3746–3758, May 2009.

[20] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Trans. Wireless Commun.*, vol. 1, no. 4, pp. 660–670, Oct. 2002, doi: 10.1109/TWC.2002.804190.

[21] O. Younis and S. Fahmy, "HEED: A hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks," *IEEE Trans. Mobile Comput.*, vol. 3, no. 4, pp. 366–379, Oct./Dec. 2004, doi: 10.1109/TMC.2004.41.

[22] Y. Zhu, R. Vedantham, S.-J. Park, and R. Sivakumar, "A scalable correlation aware aggregation strategy for wireless sensor networks," in *Proc. 1st Int. Conf. Wireless Internet (WICON)*, Jul. 2015, pp. 122–129, doi: 10.1109/WICON.2005.4.

[23] N. Sadagopan and B. Krishnamachari, "Maximizing data extraction in energy-limited sensor networks," in *Proc. IEEE INFOCOM*, vol. 3, Mar. 2004, pp. 1717–1727, doi: 10.1109/INFCOM.2004.1354583.

[24] F. Ordónez and B. Krishnamachari, "Optimal information extraction in energy-limited wireless sensor networks," *IEEE J. Sel. Areas Commun.*, vol. 22, no. 6, pp. 1121–1129, Aug. 2004, doi: 10.1109/JSAC.2004.830930.

[25] A. T. T. He, B. M. Blum, and S. JA, "AIDA: Adaptive application independent data aggregation in WSNs," *ACM Trans. Embed. Comput.*, vol. 3, no. 2, pp. 426–457, 2004.

[26] C. Cappiello and F. A. Schreiber, "Quality- and energy-aware data compression by aggregation in WSN data streams," in *Proc. IEEE Int. Conf. Pervas. Comput. Commun.*, Mar. 2009, pp. 1–6, doi: 10.1109/PERCOM.2009.4912866.

[27] T. A. Welch, "A technique for high-performance data compression," *IEEE Comput.*, vol. 17, no. 6, pp. 8–19, Jul. 1984.

[28] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Trans. Inf. Theory*, vol. IT-23, no. 3, pp. 337–343, May 1977, doi: 10.1109/TIT.1977.1055714.

[29] J. Ziv and A. Lempel, "Compression of individual sequences via variable-rate coding," *IEEE Trans. Inf. Theory*, vol. IT-24, no. 5, pp. 530–536, Sep. 1978.

[30] C. M. Sadler and M. Martonosi, "Data compression algorithms for energy-constrained devices in delay tolerant networks," in *Proc. 4th Int. Conf. Embedded networked sensor Syst. (SenSys)*, Oct. 2006, pp. 265–278, doi: 10.1145/1182807.1182834.

[31] K. Barr and K. Asanović, "Energy-aware lossless data compression," *ACM Trans. Comput. Syst.*, vol. 24, no. 3, pp. 250–291, 2006, doi: 10.1145/1151690.1151692.

[32] P. Elias, "Predictive coding-part–I," *IRE Trans. Inf. Theory*, vol. 1, no. 1, pp. 16–24, Mar. 1955.

[33] Y. Liang and W. Peng, "Minimizing energy consumptions in wireless sensor networks via two-modal transmission," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 1, pp. 12–18, Jan. 2010, doi: 10.1145/1672308.1672311.

[34] B. E. N. Greenstein, D. Estrin, and J. Heidemann, "Authors multiresolution storage and search in sensor networks university of Massachusetts," IEEE Comput., Sperry Res. Centre, MA, USA, Tech. Rep., 2007, no. 6.

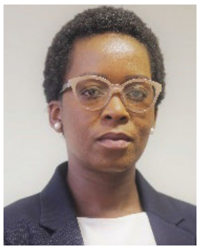
[35] Y. C. Wang, Y. Y. Hsieh, and Y. C. Tseng, "Multiresolution spatial and temporal coding in a wireless sensor network for long-term monitoring applications," *IEEE Trans. Comput.*, vol. 58, no. 6, pp. 827–828, Jan. 2009, doi: 10.1109/TC.2009.20.

[36] W. C. Li, L. M. Ang, and P. S. Kah, "Survey of image compression algorithms in wireless sensor networks," in *Proc. Int. Symp. Inf. Technol. (ITSim)*, vol. 3, Aug. 2008, pp. 1–9, doi: 10.1109/ITSIM.2008.4631875.

[37] S. S. Pradhan, J. Kusuma, and K. Ramchandran, "Distributed compression in a dense microsensor network," *IEEE Signal Process. Mag.*, vol. 19, no. 2, pp. 51–60, Mar. 2002, doi: 10.1109/79.985684.
[38] H. Amiram and T. Berger, "Rate-distortion for correlated sources with partially separated encoders," *IEEE Trans. Inf. Theory*, vol. IT-28, no. 6, pp. 828–840, Nov. 1982.
[39] R. Cristescu, B. Beferull-Lozano, and M. Vetterli, "Networked Slepian-Wolf: Theory, algorithms, and scaling laws," *IEEE Trans. Inf. Theory*, vol. 51, no. 12, pp. 4057–4073, Nov. 2005, doi: 10.1109/TIT.2005.858980.
[40] J. Chou, D. Petrovic, and K. Ramchandran, "A distributed and adaptive signal processing approach to exploiting correlation in sensor networks," *Ad Hoc Netw.*, vol. 2, no. 4, pp. 387–403, Oct. 2004, doi: 10.1016/j.adhoc.2003.09.001.
[41] K. Yuen, B. Liang, and B. Li, "A distributed framework for correlated data gathering in sensor networks," *IEEE Trans. Veh. Technol.*, vol. 57, no. 1, pp. 578–593, Jan. 2008, doi: 10.1109/TVT.2007.905243.
[42] J. B. Predd, S. R. Kulkarni, and H. V. Poor, "A collaborative training algorithm for distributed learning," *IEEE Trans. Inf. Theory*, vol. 55, no. 4, pp. 1856–1871, Apr. 2009, doi: 10.1109/tit.2009.2012992.
[43] F. Perez-Cruz and S. R. Kulkarni, "Robust and low complexity distributed Kernel least squares learning in sensor networks," *IEEE Signal Process. Lett.*, vol. 17, no. 4, pp. 355–358, Apr. 2010, doi: 10.1109/LSP.2010.2040926.
[44] J. Haupt, W. U. Bajwa, M. Rabbat, and R. Nowak, "Compressed sensing for networked data," *IEEE Signal Process.*, vol. 25, no. 2, pp. 92–101, Mar. 2008.
[45] C. Luo, F. Wu, J. Sun, and C. W. Chen, "Compressive data gathering for large-scale wireless sensor networks," in *Proc. 15th Annu. Int. Conf. Mobile Comput. Netw. (MobiCom)*, 2009, pp. 145–156, doi: 10.1145/1614320.1614337.
[46] S. Lee, S. Pattem, M. Sathiamoorthy, B. Krishnamachari, and A. Ortega, "Spatially-localized compressed sensing and routing in multi-hop sensor networks," in *GeoSensor Networks* (Lecture Notes in Computer Science), vol. 5659. Springer, Jul. 2009, pp. 11–20, doi: 10.1007/978-3-642-02903-5_2.
[47] S. Ji, Y. Xue, and L. Carin, "Bayesian compressive sensing," *IEEE Trans. Signal Process.*, vol. 56, no. 6, pp. 2346–2356, Jun. 2008, doi: 10.1109/TSP.2007.914345.
[48] J. Kho, L. Tran-Thanh, A. Rogers, and N. R. Jennings, "Decentralised control of adaptive sampling and routing in wireless visual sensor networks," *Proc. Int. J. Conf. Auton. Agents Multiagent Syst. AAMAS*, vol. 2, no. 212, pp. 1208–1209, 2009.
[49] F. Huang and Y. Liang, "Towards energy optimization in environmental wireless sensor networks for lossless and reliable data gathering," in *Proc. IEEE Int. Conf. Mobile Adhoc Sensor Syst.*, Oct. 2007, pp. 1–6, doi: 10.1109/MOBHOC.2007.4428745.
[50] Z. Tang, I. A. Glover, A. N. Evans, D. M. Monro, and J. He, "An adaptive distributed source coding scheme for wireless sensor networks," in *Proc. 12th Eur. Wireless Conf. Enabling Technol. Wireless Multimedia Commun.*, Apr. 2011, pp. 1–6.
[51] F. Oldewurtel, J. Riihijarvi, and P. Mahonen, "Efficiency of distributed compression and its dependence on sensor node deployments," in *Proc. IEEE 71st Veh. Technol. Conf.*, May 2010, pp. 1–5, doi: 10.1109/VETECS.2010.5494180.
[52] W. Wang, D. Peng, H. Wang, H. Sharif, and H. H. Chen, "Cross-layer multirate interaction with distributed source coding in wireless sensor networks," *IEEE Trans. Wireless Commun.*, vol. 8, no. 2, pp. 787–795, Feb. 2009, doi: 10.1109/TWC.2009.071009.
[53] J.-J. Xiao, A. Ribeiro, Z.-Q. Luo, and G. B. Giannakis, "Distributed compression-estimation using wireless sensor networks," *IEEE Signal Process. Mag.*, vol. 23, no. 4, pp. 27–41, Jul. 2006.
[54] A. Oka and L. Lampe, "Energy efficient distributed filtering with wireless sensor networks," *IEEE Trans. Signal Process.*, vol. 56, no. 5, pp. 2062–2075, Apr. 2008, doi: 10.1109/TSP.2007.911496.
[55] J. Teng, H. Snoussi, and C. Richard, "Decentralized variational filtering for target tracking in binary sensor networks," *IEEE Trans. Mobile Comput.*, vol. 9, no. 10, pp. 1465–1477, Oct. 2010, doi: 10.1109/TMC.2010.117.
[56] *Codes for the Manuscript*. Accessed: Sep. 3, 2021. [Online]. Available: https://github.com/zungeru01/DATA-COMPRESSION-ALGORITHMS-FOR-WIRELESS-SENSOR-NETWORKS-A-REVIEW-AND-COMPARISON-.git

KELEADILE LUCIA KETSHABETSWE (Member, IEEE) received the B.Eng. degree (Hons.) in telecommunications from the University of Essex, U.K., and the Master of Engineering (M.Eng.) degree in computer and telecommunications from Botswana International University of Science and Technology (BIUST), where she is currently pursuing the Ph.D. degree in computer and telecommunications. She worked as an Electronics Engineer at Botswana Technology Centre (BOTEC)—a research, design, and development organization, from 2000 to 2013. Before working with BOTEC, she worked as a Telecommunications Officer at the Department of Civil Aviation, responsible for maintenance and repair of telecommunications and navigational aids equipment. She is a Lab Technician with the Electronics Laboratory, BIUST. She is a Membership Development Officer with the IEEE Botswana Sub-Section Committee.

ADAMU MURTALA ZUNGERU (Senior Member, IEEE) received the B.Eng. degree from the Federal University of Technology Minna, Nigeria, the M.Sc. degree from Ahmadu Bello University Zaria, Nigeria, and the Ph.D. degree from Nottingham University, U.K. He was a Research Fellow at Massachusetts Institute of Technology (MIT), USA. He is currently working as a Professor and the Head of the Department of Electrical, Computer, and Telecommunications Engineering, Botswana International University of Science and Technology (BIUST). Before joining BIUST, in 2015, he was a Senior Lecturer and the Head of the Electrical and Electronics Engineering Department, Federal University Oye Ekiti, Nigeria. He is the Inventor of a termite-hill routing algorithm for wireless sensor networks. He has three of his patent applications registered with the World Intellectual Property Organization (WIPO). He has authored five academic books and over 60 international research articles in reputable journals, including the IEEE Systems Journal, IEEE Internet of Things Journal, IEEE Access, *JNCA* (Elsevier), and others, with over 1000 citations and an H-Index of 15. He is a Senior Member of the Institute of Electrical and Electronics Engineers (IEEE), a Registered Engineer with the Council for The Regulation of Engineering in Nigeria (COREN), and a Professional Engineer registered with Botswana Engineers Registration Board (ERB) and the Association for Computing Machinery (ACM) in the USA. He obtained the Postgraduate Teaching Certificate from MIT, in 2014. He served as the Chairperson for the IEEE Botswana Sub-Section, from 2019 to 2020. He has served as an International Reviewer for IEEE Transactions on Industrial Informatics, IEEE Sensors Journal, IEEE Access, IEEE Transactions on Mobile Computing, IEEE Transactions on Sustainable Computing, *JNCA* (Elsevier), and numerous others.



BOKANI MTENGI (Member, IEEE) received the B.Eng. degree in biomedical engineering from the University of Hartford, USA, and the M.Eng. and Ph.D. degrees in electrical engineering from Howard University, USA. She joined Botswana International University of Science and Technology as a Lecturer with the Department of Electrical, Computer and Telecommunications Engineering, in 2018. She is currently a Senior Lecturer. She worked as an Electron Microscopy Imaging Engineer at Howard Hughes Medical Research Lab, from 2016 to 2018. She was employed as a Biomedical Engineer by the Ministry of Health, Botswana, from 2004 to 2007. Her teaching and research interests include power electronics, energy systems, and bioelectronics. She is the current Secretary of the IEEE Botswana Subsection and the Country Representative of the IEEE Women in Power (WiP) Southern Region.

CASPAR K. LEBEKWE (Member, IEEE) received the M.Eng. degree in electronics and communications engineering from the University of Bath, in 2008, and the Ph.D. degree in electrical and electronics engineering from the University of Bath, sponsored by the General Lighthouse Authorities. His Ph.D. project was focused on eLoran service volume coverage prediction. He is currently a Lecturer at Botswana International University of Science and Technology, where he teaches optical communications, antennas and propagation, discrete mathematics, telemetry, and remote control as well as electromagnetic field theory. He is a Registered Member of IEEE.

S. R. S. PRABAHARAN (Member, IEEE) was the Dean of School of Electronics Engineering (SENSE), VIT University, Chennai Campus, from August 2015 to August 2018. He has worked with Manipal International University, Malaysia, the University of Nottingham, Malaysia/U.K. campuses, and the CSIR National Laboratories (NAL and CECRI). He spent significant time in France and USA labs and universities as a Postdoctoral Fellow and a Technology Consultant. He has served as a Faculty Member for various international universities, including the University of Malaya, Universiti Teknologi Petronas, Multimedia University, UPMC, Paris, UPJV, Amiens, France, and Tohoku University, Japan. He has also worked as an Adjunct Professor with Malaysia University of Science and Technology and Manipal International University. Recently, he has held a visiting professorship at Tohoku University. He served as a Consultant to the battery and supercapacitor industries. He is currently a Full Professor and the Joint Director of the Directorate of Research, SRM Institute of Science and Technology, Chennai, India. He has received several research grants from DST, DRDO, Young Scientist Grant Scheme, Malaysian Government Grant (MOSTI and MOHE), and Motorola Inc., USA. His expertise focuses on supercapacitors for a variety of applications, including naval missiles (DRDO). He has published over 75 high impact journal articles and over 200 conference papers and holds international granted patents (Malaysia, USA, and Japan). He is known for his notable research in the areas of power sources (Li-Ion batteries and supercapacitors), RF Energy harvesting, advanced power solutions for IoT systems, and memristors. His current research interests include memristors and neuristors. He has served for various expert committees in Malaysia (Multimedia Development Corporation and Office of Prime Minister Department (UNIK) and FP7 European Union Committee). He was the Technical Consultant to United States Army Research Project under Southern University, Baton Rouge, USA. He was a recipient of the Young Scientist Award, in 1994. He was a recipient of the prestigious Erasmus Mundus Visiting Scholar under the European Union (EU) and held a visiting professorship in France. He was also a recipient of U.K.-Singapore Visiting Scientist Award in The University of Sheffield, U.K., in 2007. He has delivered over 52 invited/keynote talks in international scientific conferences/symposia. He has edited a few research books. He has been a guest editor for international journals as well.