Journal of Informatics and Web Engineering
Vol. 3 No. 1 (February 2024) eISSN: 2821-370X

Optimizing Medical IoT Disaster Management with Data Compression

Nunudzai Mrewa[1], Athirah Mohd Ramly[1,2,3], Angela Amphawan[1,2], Tse-Kian Neo[4*]
[1]Department of Computing and Information Systems, School of Engineering, Sunway University, 47500 Petaling Jaya, Malaysia.
[2]Smart Photonics Research Laboratory, School of Engineering and Technology, Sunway University, 47500 Petaling Jaya, Malaysia
[3]Research Centre for Human-Machine Collaboration (HUMAC), Department of Computing and Information Systems, Sunway University, 47500 Petaling Jaya, Malaysia.
[4]CAMELOT, Faculty of Creative Multimedia, Multimedia University, Cyberjaya 63100 Selangor, Malaysia
Corresponding author: (tkneo@mmu.edu.my; ORCiD: 0000-0002-5991-7409)

Abstract - In today's technological landscape, the convergence of the Internet of Things (IoT) with various industries showcases the march of progress. This coming together involves combining diverse data streams from different sources and transmitting processed data in real-time. This empowers stakeholders to make quick and informed decisions, especially in areas like smart cities, healthcare, and industrial automation, where efficiency gains are evident. However, with this convergence comes a challenge – the large amount of data generated by IoT devices. This data overload makes processing and transmitting information efficiently a significant hurdle, potentially undermining the benefits of this union. To tackle this issue, we introduce "Beyond Orion," a novel lossless compression method designed to optimize data compression in IoT systems. The algorithm employs advanced techniques such as Lempel Ziv-Welch and Huffman encoding, while also integrating strategies like pipelining, parallelism, and serialization for greater efficiency and lower resource usage. Through rigorous experimentation, we demonstrate the effectiveness of Beyond Orion. The results show impressive data reduction, with up to 99% across various datasets, and compression factors as high as 7694.13. Comparative tests highlight the algorithm's prowess, achieving savings of 72% and compression factor of 3.53. These findings have significant implications. They promise improved data handling, more effective decision-making, and optimized resource allocation across a range of IoT applications. By addressing the challenge of data volume, Beyond Orion emerges as a significant advancement in IoT data management, marking a substantial step towards realizing the full potential of IoT technology.

Keywords—Data Compression, Internet of Things (IoT), Disaster Management, Enhanced Lempel Ziv-Welch and Huffman, Pipelining, Parallelism, Serialization

Received: 29 May 2023; Accepted: 12 September 2023; Published: 16 February 2024

I. INTRODUCTION

The convergence between the Internet of Things (IoT) and various industries has brought about significant advancements in connectivity and data exchange as defined by [1]. IoT refers to a system of interconnected physical devices, vehicles, buildings, and objects that collect and exchange data over the internet [2]. By embedding computing devices with sensors, software, and communication technologies, IoT enables seamless communication and automation, revolutionizing industries such as smart cities, healthcare, and industrial automation [3]. The potential applications of IoT are vast and diverse. In smart cities, IoT enables efficient infrastructure management, optimized transportation systems, and enhanced public safety through interconnected devices and data driven decision-making. In healthcare, IoT facilitates remote patient monitoring, personalized

Journal of Informatics and Web Engineering
https://doi.org/10.33093/jiwe.2024.3.1.4

Published by MMU Press. URL: https://journals.mmupress.com/jiwe

treatment, and improved access to medical services. Industrial automation benefits from IoT by enabling real-time monitoring, predictive maintenance, and enhanced operational efficiency as mentioned by [4]. The immense potential of IoT is evident in predictions by [5] which estimates that by 2030, there will be 500 billion connected devices worldwide. However, alongside the opportunities presented by this widespread adoption of IoT, there are implementation challenges that must be addressed. One of the prominent challenges associated with the deployment of IoT devices is the management of large volumes of real-time generated data is what [6] concluded in their research. This poses significant computational and bandwidth limitations, which directly impact the efficiency of data processing and transmission. In resource-critical scenarios like disaster monitoring and patient care, the ability to make instant decisions based on this data becomes crucial. However, the scale of data generated poses constraints that need to be addressed for seamless operation and optimal performance of IoT devices.

Research by [6] also illustrates that, IoT devices face energy wastage due to idle listening, overhearing, packet overload, frequent switching, and over-emitting. These factors contribute to unnecessary energy consumption, collisions, higher data rates, and information loss. Resolving these issues is crucial for optimizing energy usage and improving overall system performance in IoT applications. In this paper, we will review the existing literature to provide a comprehensive overview of the current state of research in IoT data processing and compression techniques. We will examine the implications and limitations of current approaches, identifying the gaps in the literature and establishing the need for innovative solutions. Additionally, we will present the Beyond Orion compression scheme, a novel lossless compression algorithm developed from Lempel-Ziv, Huffman, and LZ-Welch techniques. We will evaluate the performance of Beyond Orion through comprehensive experimentation and analysis, demonstrating its impressive data reduction capabilities. Furthermore, we will highlight the potential of Beyond Orion in expediting data collection, improving decision-making, and optimizing resource utilization in various IoT applications

II. LITERATURE REVIEW

Data compression serves the purpose of converting original information into a more compact format, facilitating efficient transmission and storage. Its origins can be traced back to Samuel Morse's creation of Morse code in 1838 as documented by [7]. [7] also states that Morse code utilized dots and dashes to represent letters, employing shorter sequences for frequently occurring letters. This ingenious approach minimized message size and transmission time. Since then, compression schemes have evolved, falling into two broad categories: lossy and lossless. In the existing literature on data compression, two main approaches are commonly discussed: lossless compression and lossy compression. Lossless compression techniques aim to reduce the file size while preserving all the information required for the reconstruction of the original data. On the other hand, lossy compression techniques permanently reduce the file size by eliminating redundant data, but at the cost of irretrievable loss of some information. It is important to note that in lossy compression, the removal of redundant data may impact the functionality of certain applications.

Considering the specific context of the Internet of Things (IoT) environment, where data processing and decision making rely on the integrity of every bit of information, the use of lossless data compression becomes highly desirable. By ensuring that all bits are preserved, lossless compression techniques allow for accurate data analysis and reliable decision-making processes within IoT systems. It is worth mentioning that data can be characterized in various ways, leading to the development of diverse data compression techniques. Researchers have proposed different algorithms and methods to address the specific needs and challenges of IoT data compression. Hence, it becomes crucial for future researchers to evaluate and compare the existing methods in order to choose the most suitable algorithms that can effectively enhance the efficiency of IoT devices.

An approach utilizing a simple data compression algorithm, as proposed by [8], addresses the challenges of memory usage and computational complexity in IoT devices, specifically focusing on developing a tailored lossless compression technique for IoT nodes. The research conducted an analysis on Motiev's Tmote sensor nodes to evaluate the impact of their compression scheme. The findings of the study demonstrated promising results in terms of compression ratios. The proposed algorithm achieved compression ratios of 66.69% and 76.33%, which significantly outperformed other compression techniques such as S-LZW, gzip, and bzip2. These results indicate the high performance of the compression algorithm in reducing the memory usage of IoT nodes. However, there are certain aspect that require further investigation. First, the research did not extensively explore the energy consumption and computational complexity of the proposed algorithm. It is crucial to assess these factors to gain a comprehensive understanding of the algorithm's overall efficiency and feasibility in real-world IoT deployments. Moreover, the study did not evaluate the energy consumption during the research, which represents an important consideration in IoT environments where energy efficiency is a critical factor. Further

research should investigate the energy consumption of the proposed algorithm to determine its impact on the overall energy consumption of IoT nodes. Additionally, the study compared the proposed compression algorithm with techniques that are not commonly used within IoT nodes due to their computational complexity. To provide a more accurate evaluation of compression performance in the IoT domain, future research should focus on assessing the performance of compression standards specifically tailored for IoT applications.

The research conducted [9] developed an adaptive compression scheme using S-LEC (sequential lossless entropy scheme) and S-LZW (sensor Lempel–Ziv–Welch) data compression schemes to reduce the total traffic power consumption in the cloud-based IoT network. The proposed scheme has the ability to choose the most energy-efficient data compression method, taking into account factors such as the battery level and processing capability of the IoT device. In comparison to a non-compression scheme, the adaptive algorithm demonstrated energy savings of 33% and 40%. These savings can be attributed to a decrease in the number of hops and traffic load within the network, which enables the system to effectively handle increased traffic demand and prolong the lifespan of IoT devices by 50%. In a separate study by [10] conducted an investigation into the performance of an algorithm based on the Lempel-Ziv-Welch (LZW) technique. While the statistical variable-length Huffman method achieved a compression rate of 20% for text, the LZW algorithm demonstrated a compression range of 40% to 60% for data. The research highlights the robustness of the LZW algorithm in compression and emphasizes the improvement in efficiency with larger codes. Both [9] [10] shed light on the potential of employing adaptive compression range.

The study conducted by [11] focused on the analysis and implementation of the DCT (Discrete Cosine Transform) and DWT (Discrete Wavelet Transform) algorithms using the Tiny Operating System on the TelosB hardware platform. The experiment aimed to transmit multiple JPEG images between nodes and evaluate the compression results when the images were reconstructed during transmission. The performance of the compression techniques was assessed based on several metrics, including the signal-to noise ratio (PSNR), throughput, compression ratio, end-to-end delay, and battery lifetime. The results indicate that the DWT algorithm outperformed the DCT algorithm in terms of speed and energy consumption. This suggests that DWT is a more efficient compression technique for image compression. However, when the topology was changed and more intermediate nodes were introduced, both algorithms exhibited a deterioration in performance. This finding implies that these algorithms may not perform well in large-scale applications. Further investigation is needed to understand the impact of changing the topology in greater detail and to identify an algorithm that can deliver efficient data transmission regardless of external changes. Moreover, it is recommended to conduct further research in a more realistic environment to analyze the performance of the algorithms on a larger scale. Additionally, implementing the algorithms on other platforms can provide insights into their performance characteristics and suitability for different hardware configurations. These additional investigations can contribute to a more comprehensive understanding of the algorithms' capabilities and limitations.

The research by [12] conducted a study to explore a computationally lightweight algorithm for IoT devices derived from the LZW (Lempel Ziv Welch) scheme. The experiment was performed on a MICAz system, which operates on the Tiny Operating System specifically designed for devices with limited resources. The study utilized temperature readings collected over a period of 7 days from sensors. The performance evaluation of the algorithm focused on two key metrics: compression ratio and data communication energy. The results demonstrated that the algorithm achieved an energy efficiency of up to 85%, indicating its potential for reducing energy consumption in IoT devices. This highlights the effectiveness of the LZW algorithm as a lossless compression scheme for IoT applications. However, it is important to note that the study primarily focused on a specific dataset and operating conditions. Additional research is required to evaluate the algorithm's robustness and effectiveness in various scenarios, such as different data volumes and transmission frequencies.

The algorithm known as Robust Information-Driven Architecture (RIDA), proposed by [13] aims to enhance compression by identifying correlations among data from groups of sensors. The authors of the algorithm make the assumption that if two nodes within the same cluster need to communicate, only a single hop will be required. RIDA comprises three main components: information-driven logical mapping, resiliency mechanism, and compression algorithm. In the information-driven logical mapping phase, nodes within the cluster exchange their readings and collectively learn patterns from the entire cluster. Once a pattern is identified, the resiliency mechanism detects any faulty nodes and isolates them from the rest of the nodes. The compression algorithm is then applied to the data. The researchers conducted tests on publicly accessible real-world datasets, utilizing both the Discrete Cosine Transform (DCT) and the Discrete Wavelet Transform (DWT) in the architecture. The results demonstrated that in ordinary multi-hop data networks, RIDA achieved energy savings of 30% and bandwidth savings ranging from 80% to 95%. Furthermore, after decompression, the original data could be restored with a low error rate of approximately 3%. However, it is important to note that these results may not be directly applicable to networks of mobile sensors, and the effectiveness of this approach in high data rate compression for

audio and video data remains unclear. Further research and experimentation would be necessary to evaluate the performance of RIDA in such scenarios.

In the study presented in [14] the authors proposed a dynamic lossy compression method called Senscompr. Their approach involved reconstructing various sensor data using the Chebyshev approximation model. The Chebyshev approximation model helps eliminate redundant data. Additionally, the algorithm introduced dynamic block sizes as a solution to the limitations of fixed block sizes commonly used in compression methods. In another research work by [15], the focus was on addressing storage and precision issues in video streams. The authors employed a metric called Structural Similarity Index Measure (SSIM) to evaluate the redundancy of video frames. Only the frames that differed significantly from each other were stored, while redundant frames were discarded. The results of their study showed a 40% reduction in video size compared to the original uncompressed video. These studies highlight the effectiveness of dynamic compression techniques in reducing data size and eliminating redundancy.

The Senscompr algorithm proposed in [14] leverages the Chebyshev approximation model, while the study in [15] utilizes the SSIM metric for video compression. To accurately assess energy efficiency in data compression, factors such as battery consumption rate and battery life need to be considered. These criteria provide insights into the energy consumption patterns of devices or nodes within an IoT network. By implementing data compression at the end nodes of the network, it is possible to improve battery consumption across the entire network, distribute the computational load between devices, and extend the overall battery life. A more comprehensive evaluation of data compression standards should include an analysis of energy consumption patterns, battery consumption rates, and the impact of compression algorithms on the battery life of IoT devices. By considering these factors, researchers can provide a more detailed understanding of the energy efficiency implications of different data compression techniques. This, in turn, can contribute to the development of more energy-efficient IoT systems and enhance the overall sustainability of IoT deployments. In conclusion, the literature review highlights the challenges associated with selecting the appropriate compression technique for low bandwidth IoT devices. The conventional approaches such as Huffman coding, Entropy coding, bit plane coding, and run-length coding exhibit limitations in terms of memory usage and computing power, making them unsuitable for resource-constrained IoT networks. The Z standard and dictionary-based techniques such as LZO, LZW, and LZ4 offer promising solutions due to their trade-offs between compression rates and ratios, as well as their computational efficiency. However, to address the limitations and enhance compression performance, a more comprehensive evaluation of these algorithms is required. Additionally, adopting a hybrid approach that combines the benefits of multiple compression techniques can offer a versatile tool for efficient data compression in low-bandwidth IoT environments.

III. RESEARCH METHODOLOGY

The Beyond Orion algorithm, also known is a data compression technique that combines the concepts of Lempel-Ziv77 compression, Huffman Coding, and Lempel-Ziv-Welch (LZW) compression. The algorithm aims to efficiently compress data by identifying repeated sequences, encoding them using variable-length codes, and further compressing the resulting code dictionaries. To begin with, the algorithm utilizes the Lempel-Ziv77 approach to discover repeated sequences within the data stream. These sequences are represented as triplets, following the format. The" offset" denotes the number of steps required to traverse back in the data stream to locate the beginning of the current sequence. The" length" indicates the number of steps to progress from the current position to encompass the identified repeated sequence. Lastly, the" literal" refers to the value that immediately follows the repeated sequence.

Upon the construction of these triplets, the algorithm proceeds to employ Huffman Coding to encode the shorter lists that represent the data stream. Huffman Coding generates code dictionaries, which contain mappings between the sequences and their respective variable-length codes. This step facilitates the compression of the data by assigning shorter codes to frequently occurring sequences and longer codes to less frequent ones, thereby achieving an optimal balance between compression efficiency and decoding complexity. To further compress the data, the LZW algorithm, specifically the lzw compress function, is employed. This phase compresses the code dictionaries generated by Huffman Coding, reducing their size and enhancing the overall compression ratio. The compressed data is subsequently stored as a tuple consisting of three integers (offset, length, and literal) and a dictionary featuring three keys (offset, length, and literal). By utilizing integer representations for the values, instead of string-based representations, the algorithm achieves better space-saving capabilities.

Suppose the Beyond Orion algorithm is employed to compress the phrase "The cat chased the cat in the garden." The algorithm utilizes the Lempel-Ziv77 approach to identify repeated sequences within the data stream. These sequences are represented as triplets in the format. The "offset" signifies the number of steps required to traverse

back in the data stream to locate the beginning of the current sequence. The “length” denotes the number of steps needed to progress from the current position to encompass the identified repeated sequence. Lastly, the “literal” refers to the value that immediately follows the repeated sequence. For example, one of the triplets in the compressed dictionary could be . Here, the ”10” indicates that the sequence” cat” was encountered 10 steps back in the data stream. The ”3” signifies that the sequence spans 3 steps, including the current position. The” in the garden.” represents the literal that follows the repeated sequence. To perform LZW compression, unique codes are assigned to each dictionary entry. For instance, the compressed dictionary could include the following entries: Compressed Dictionary:

• Code: 1, Triplet: <10,3, “in the garden.”>

• Code: 2, Triplet: < 4,3, “chased the”>

• Code: 3:Triplet: <0,3, “The”>

In this example, the dictionary entries are assigned numeric codes. The compressed data consists of the triplets and the compressed dictionary. The compressed data for the given phrase includes the triplets representing the repeated sequences along with the compressed dictionary. The triplet <10,3, “in the garden.”> can be represented by the code ”1”, < 4,3, “chased the”> by ”2”, and < 0,3, “The”> by ”3”. By using these codes, the original phrase is effectively represented and compressed, resulting in improved storage efficiency. By employing the Beyond Orion algorithm and subsequent LZW compression, the original phrase “The cat chased the cat in the garden.” can be efficiently compressed using triplets and a compressed dictionary, achieving enhanced storage efficiency.

Note: The actual codes assigned to the dictionary entries may vary depending on the implementation and specific compression scenario. The Beyond Orion compression algorithm maps codes to values in a dictionary as it reads data from the stream. It constructs phrases and character sequences, assigning them codes in the dictionary. The algorithm can rebuild the dictionary during decompression. Beyond Orion is well-suited for compressing data streams due to its character-by character approach. During decoding, the dictionary is recreated using ASCII codes, and compressed data is translated back to its original text representation using the code dictionary.

A. Performance Metrics

i. Saving Percentage

Saving percentage is a metric that measures the percentage reduction in file size achieved after compression. It provides an indication of space savings resulting from the compression operation. A higher saving percentage implies better algorithm performance. The calculation for the saving percentage according to [17] is as follows:

$$\text{Saving percentage} = \left(\frac{\text{Uncompressed file size} - \text{Compressed file size}}{\text{Uncompressed file size}}\right) \times 100$$

ii. Compression Factor

It represents the ratio between the size of the original data and the size after compression. A higher compression factor indicates more efficient compression. The calculation for compression factor according to [18] is as follows:

$$\text{Compression factor} = \frac{\text{Uncompressed file size}}{\text{Compressed file size}}$$

iii. Compression and decompression time

Compression time is the duration of compression and decompression time is the duration of decompression. The time library in Python will serve as a reliable measurement tool to gauge the elapsed time for these operations.

B. Testing Scenarios

In this section, we outline the testing scenarios conducted to evaluate the performance of the Beyond Orion compression algorithm. The tests assess the robustness and reliability of the algorithm when compressing data of varying sizes. Two testing scenarios, the Standalone Test and the Comparative Test, were performed.In the Standalone Test, the Beyond Orion algorithm was independently tested in three iterations to measure its performance. Data frames of varying sizes were created, ranging from 12 Megabytes (MBs) in the first iteration then 120 megabytes in the second iteration and 1.2 Gigabytes (1200 MBs) in the third iteration. The data frames

were compressed, saved to text files, and then decompressed. Various metrics, including compression and decompression times, compression and decompression speeds, saving percentage and compression factor were calculated and analyzed. The results obtained from these calculations provided insights into the performance of the Beyond Orion algorithm.

The Comparative Test aimed to compare the performance of the Beyond Orion algorithm against three other commonly used compression algorithms: LZ4, Zlib which is an abstraction fo the deflate algorithm, and LZMA an algorithm that is part of the Lempel Ziv family (Lempel-Ziv-Markov chain). The dataset used for this test comprised patient readings obtained from various hospitals worldwide, with a total size of 174 MBs. The results obtained from each compression algorithm were carefully examined and compared. By conducting these testing scenarios, we aimed to thoroughly evaluate the performance of the Beyond Orion compression algorithm in terms of compression and decompression times, speed, saving percentage and compression factor. The results obtained from these tests will provide insights into the effectiveness and efficiency of the Beyond Orion algorithm compared to other widely used compression algorithms.

IV. RESULTS AND DISCUSSION

When comparing the compression sizes relative to the original file size of 174.43 MBs, the Zlib algorithm achieved a compressed file size of 65.29 MBs, resulting in a compression factor of 2.67 which is highlighted in figure 1. LZMA had a compressed file size of 50.28 MBs, with a compression factor of 3.47. LZ4 resulted in a compressed file size of 113.22 MBs, corresponding to a compression factor of 1.54. Lastly, Beyond Orion achieved a compressed file size of 49.40 MBs, yielding a compression factor of 3.53. These compression factors provide an indication of the level of compression achieved by each algorithm in relation to the original file size. The findings comparing the compression sizes relative to the original file size of 174.43 MBs are summarized in figure 1 below.

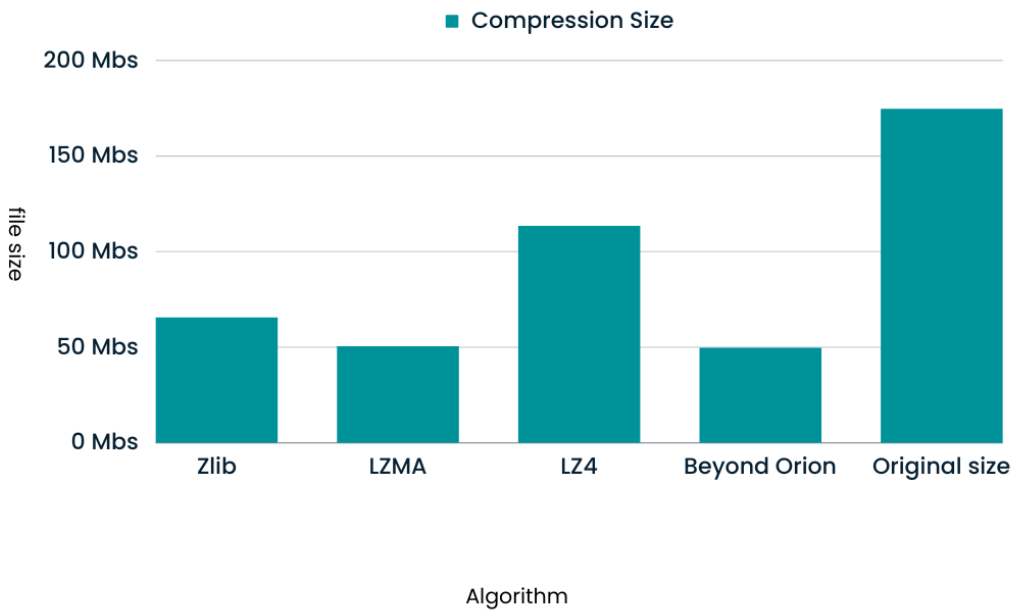

Figure 1. Performance of Compression Algorithms relative to Original

The results of figure 2 reveal the saving percentage, Zlib achieved a 63% reduction in size, LZMA achieved 71%, LZ4 achieved 35%, and Beyond Orion achieved 72%. These percentages reflect the amount of space saved by each compression algorithm. The bar graph in figure 3 indicate the compression factor. Zlib was 0.37, for LZMA it was 0.29, for LZ4 it was 0.65, and for Beyond Orion it was 0.28. These ratios provide an indication of the effectiveness of the compression algorithms in reducing the file size. When assessing the compression and decompression time, figure 4 reveals that Zlib had a compression time of 13.43 seconds and a decompression time

of 4.34 seconds. LZMA had a compression time of 95.66 seconds and a decompression time of 26.55 seconds. LZ4 had a compression time of 5 seconds and a decompression time of 2.8 seconds. Beyond Orion had a compression time of 68.75 seconds and a decompression time of 118.17 seconds. These times reflect the speed at which the algorithms perform compression and decompression operations.

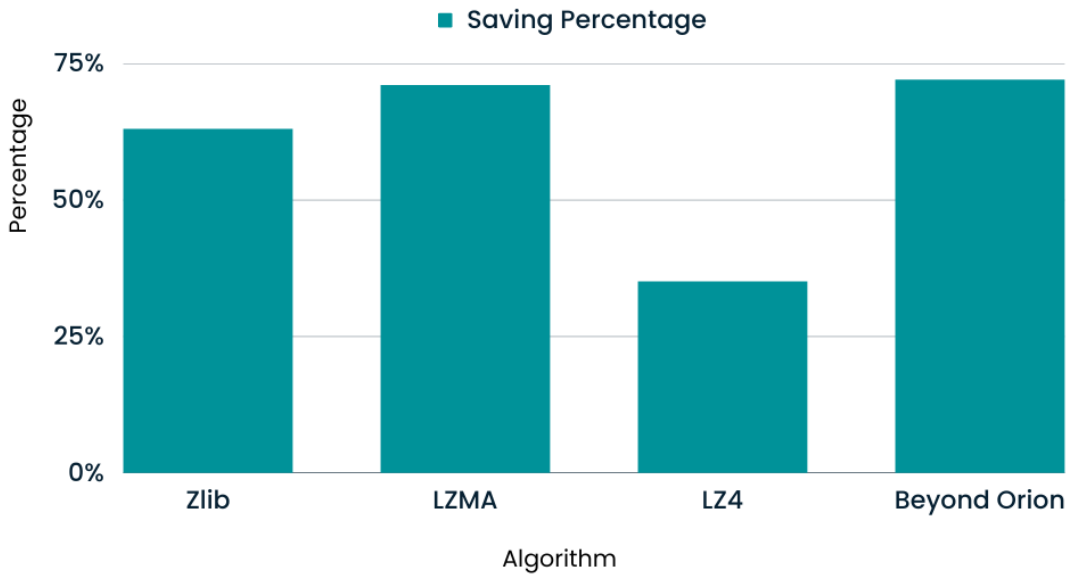

Figure 2. Saving Percentage of the Algorithms

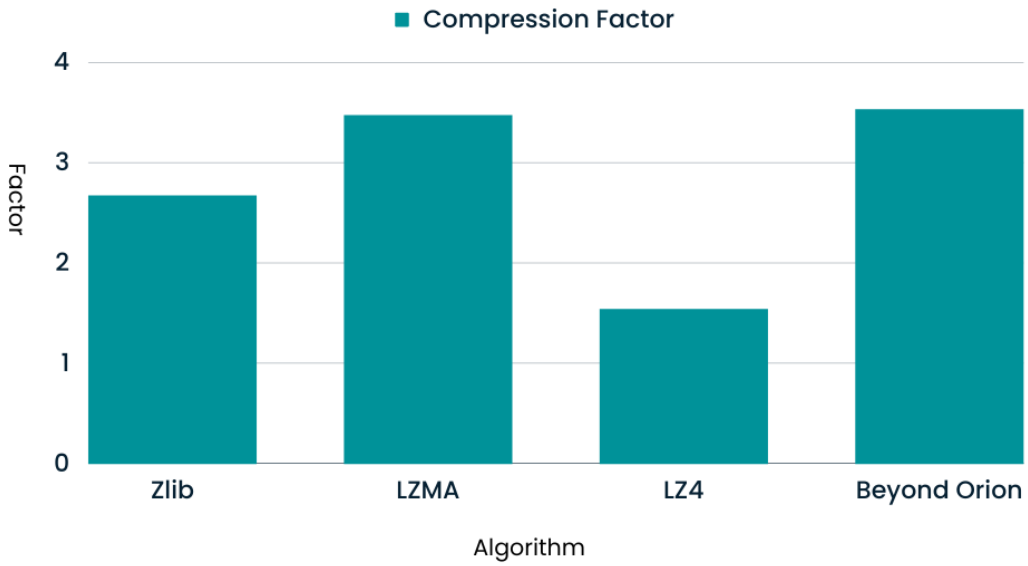

Figure 3. Compression Factor of the algorithms

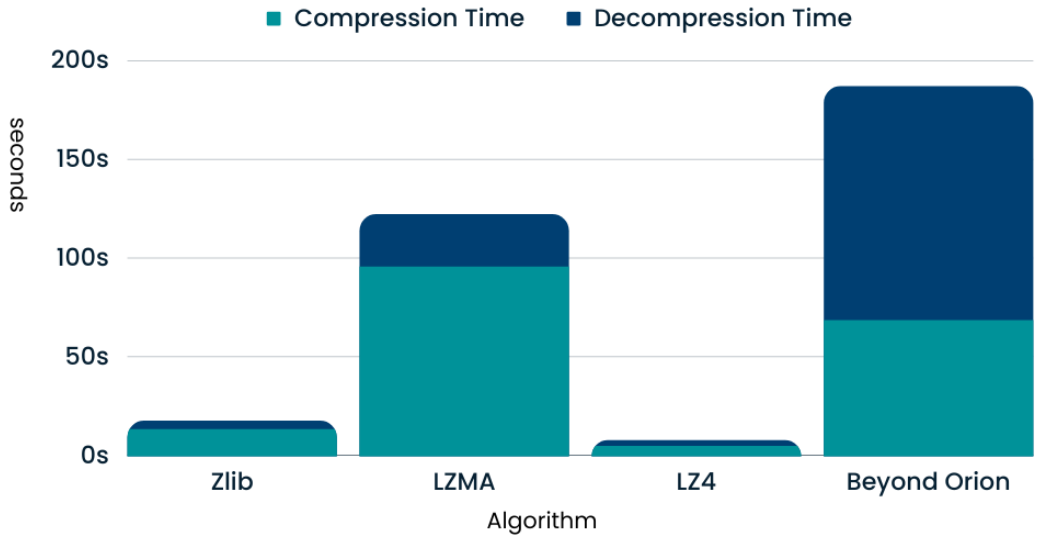

Figure 4. Compression and Decompression time of each algorithm

In summary, the Zlib algorithm achieved a good compression factor and saving percentage, with relatively fast compression and decompression times. LZMA provided higher compression and saving percentages but at the cost of longer compression and decompression times. LZ4 had lower compression and saving percentages, but faster compression and decompression times. Beyond Orion achieved the highest compression factor and saving percentage, but with longer compression and decompression times. These results demonstrate the trade-offs and performance characteristics of each compression algorithm. Commencing the initial iteration in the standalone test, which entailed a data frame size of 12 MBs, the compressed file size achieved was a mere 0.00156 MBs as noted by figure 7. Notably, the compression process was completed in 0.48 seconds, while the subsequent decompression step required 0.17 seconds as shown in figure 6. Figure 6 also demonstrates a remarkable compression factor of 7694.13, these results indicate a significant level of compression attained. Furthermore, the saving percentage reached an impressive 99.98%, underscoring the substantial reduction in file size achieved. Advancing to the second iteration, involving a larger data frame size of 120 MBs, the compressed file size remained unchanged at 0.00156 MBs.

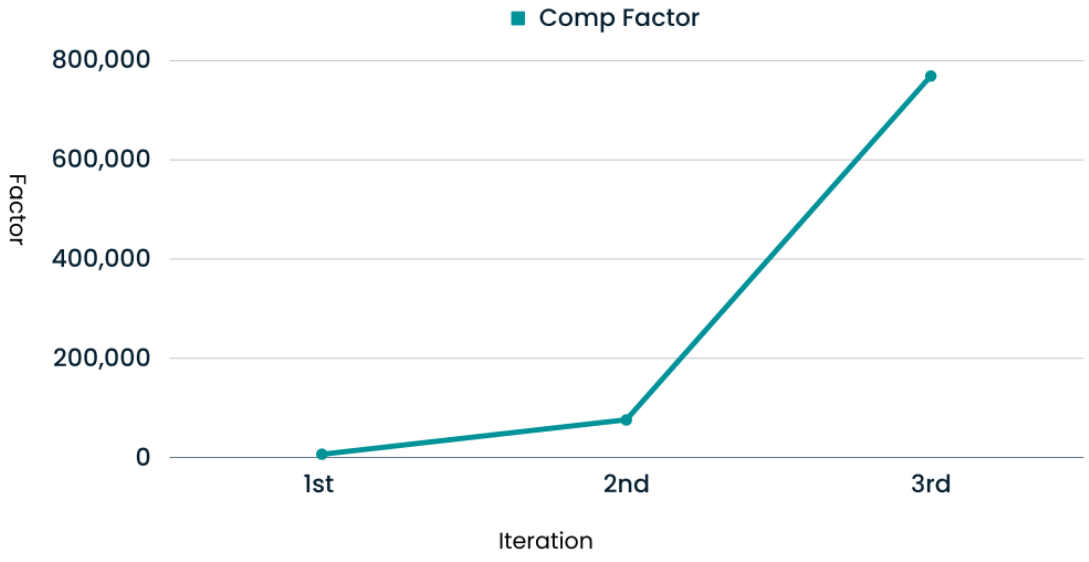

Figure 5. Compression Factor of each iteration

However, the compression time extended to 4 seconds as shown in figure 6, whereas the decompression time decreased to 0.09 seconds. These adjustments resulted in a higher compression factor of 76924.90, signifying a further enhancement in the compression achieved when compared to the first iteration shown in figure 5. Impressively, the saving percentage remained consistent at 99.98%, affirming the continued substantial reduction in file size. Correspondingly, the compression factor increased to 1.29, reflecting the compressed file size relative to the original size. In the final iteration, the data frame size surged to 1200 MBs, while the compressed file size persisted at 0.00156 MBs, underscoring the algorithm's efficiency in maintaining compression size consistency across varying data frame sizes. Nevertheless, the compression time notably increased to 67 seconds, reflecting the greater computational requirements for larger file sizes. Similarly, the decompression time was extended to 1.3 seconds. Notably, the compression factor demonstrated a significant boost, reaching 769232.59, highlighting the algorithm's remarkable ability to achieve higher levels of compression as the file size increases. The saving percentage depicted in figure 8 remained exceptionally high at 99.99% and these results demonstrate that the compression algorithm effectively reduces the file size across different data frame sizes, achieving high compression factors and saving percentages.

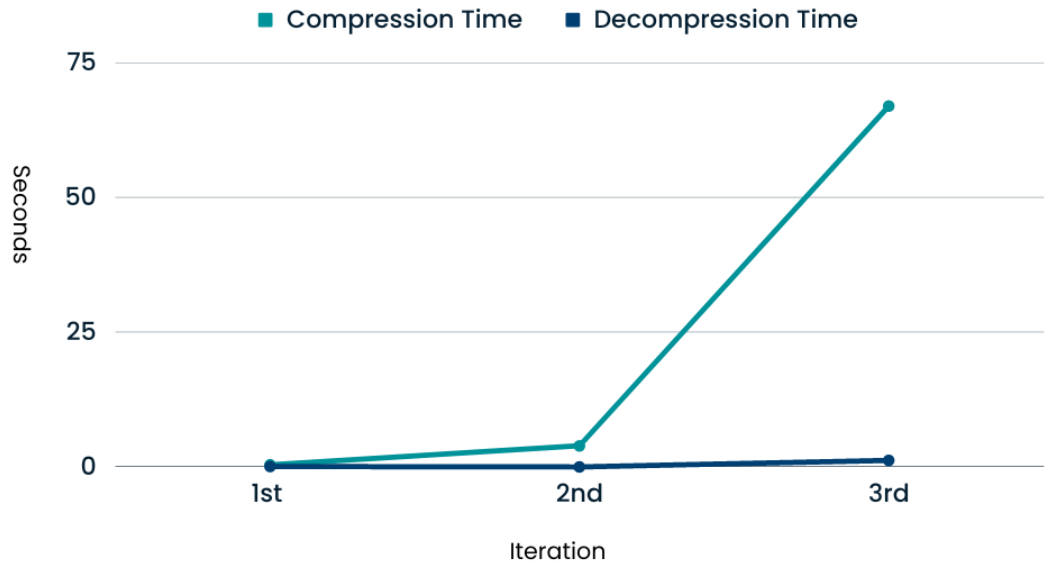

Figure 6. Compression and decompression time

In the standalone compression test, a notable finding emerged regarding the Beyond Orion algorithm's performance. Figure 7 showed a clear trend that indciated the compressed file size remained consistent across different file sizes. For instance, when compressing a 12 MB file, the resulting compressed file size was 1.56 kilobytes, which is approximately a thousand times smaller. This pattern persisted as the original file size increased to 120 MB and 1.2 Gigabytes (GBs), with the compressed file sizes becoming a thousand times and one hundred thousand times smaller, respectively. This consistency suggests that the Beyond Orion algorithm demonstrates remarkable resilience and effectiveness in handling large volumes of data.

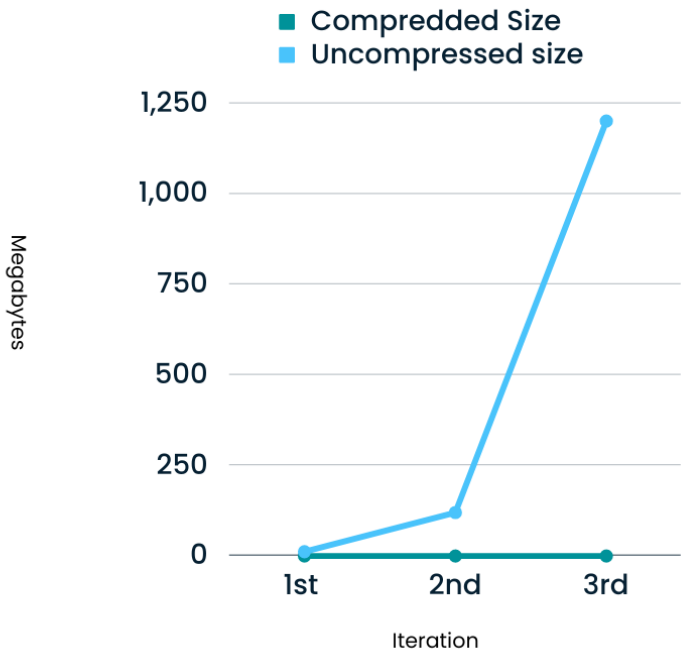

Figure 7. Compressed file size and original file size for each iteration

Similar conclusions can be drawn from the comparative test, where the Beyond Orion algorithm achieved the smallest compressed file size of 49 MBs, followed by LZMA at 50 MBs which is presented in figure 1. This finding indicates that the Beyond Orion algorithm outperforms other compression algorithms in terms of efficiency. An intriguing observation relates to the saving percentage observed in both tests. In the standalone test, a consistent saving percentage of 99% was achieved, while in the comparative test, the saving percentage reached 71%, as illustrated in Figure 2 and 8. This implies that as the file size increases, the saving percentage also increases. This behavior can be attributed to the effective utilization of dictionary encoding and statistical techniques employed by the algorithm, which excel at identifying and compressing redundancies. Consequently, larger datasets exhibit higher saving percentages as more redundant data can be compressed together.

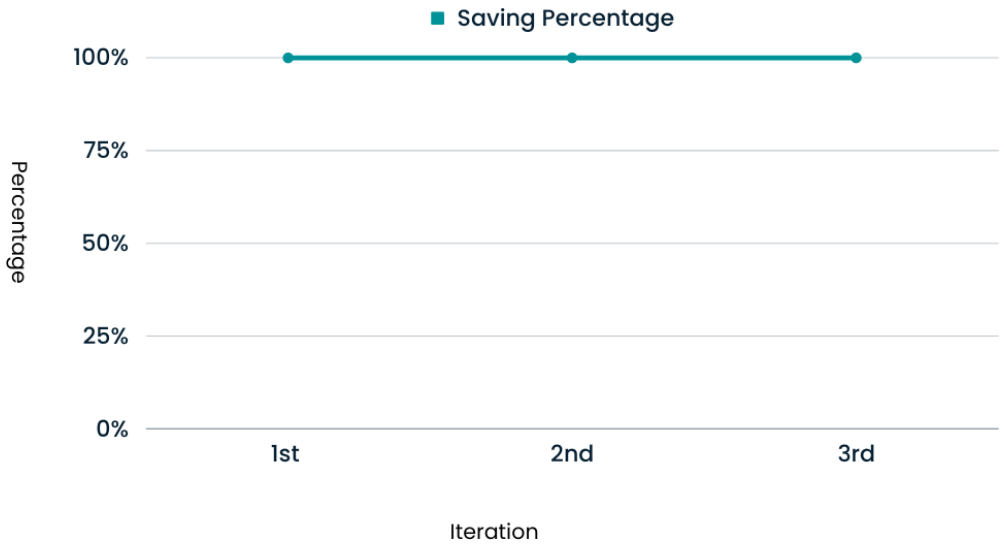

Figure 8. Saving percentage for each iteration

Considering the compression factor, the Beyond Orion algorithm attained a factor of 3.53 in the comparative test and an impressive value of 7496.13 in the standalone test as shown in figure 3 and 5. Notably, lossy compression algorithms can introduce data reconstruction errors and computational complexity. Moreover, the Beyond Orion algorithm showcased superior performance by avoiding reconstruction errors, employing lossless compression techniques that preserve all data, and maintaining manageable computational complexity compared to other algorithms. These findings demonstrate the significant advantages and potential of the Beyond Orion algorithm in the field of data compression, especially when operating with large datasets and low-power IoT environments.

V. CONCLUSION

Despite the promising performance of the Beyond Orion compression algorithm, there are a few limitations and areas for further investigation that should be acknowledged. One notable limitation is the relatively slow compression time exhibited by the Beyond Orion algorithm. In the standalone test, the compression time increased by 733.33% from the second to the third iteration, and by 1575% from the second to the third iteration as visualized in figure 4 and 6. From the data in figure 4, the algorithm took 186 seconds to perform compression and decompression. These results indicate that as the file size increases, the compression and decompression time also increases. In IoT environments where real-time data transmission and processing are crucial, a compression time of 3 minutes is impractical and would introduce significant delays. To overcome this limitation, a possible solution could involve implementing a hierarchical architecture wherein nodes are organized in clusters and relay nodes are used to perform compression and decompression tasks.

By overcoming the compression and decompression tasks to relay nodes, sensors would only be responsible for taking readings, thus reducing the overall compression and decompression time. Research by [19] and [20] suggests that the implementation of relay nodes can improve energy efficiency and data collection by acting as load balancers in the network, making them suitable for low-power IoT nodes. Furthermore, according to a study by [21], the optimal placement of relay nodes is found to be in the central position between the source and destination. The research also indicates that increasing the number of relay nodes can amplify the received signal and effectively retransmit it to the destination node. This amplification provided by relay nodes proves highly advantageous in reducing error rates. Additionally, these relay nodes can incorporate larger buffers, allowing for the compression and decompression of packets in a sequential manner, resulting in reduced overall network load and improved network performance. Further investigation is needed to address various questions raised by this research. For instance, it is important to determine the extent to which compression algorithms influence energy saving and energy usage. While it is assumed that compressing data leads to energy savings due to reduced data transmission, the implementation of energy-efficient algorithms does not guarantee an extended battery lifetime of a device [9] argues.

Therefore, it is essential to measure the battery consumption with and without compression to quantify the energy efficiency gained from using a compression algorithm. Furthermore, conducting real-world experiments would provide more tangible results and validate the proof of concept. However, it should be noted that real-world experiments may not fully replicate all aspects of the real-world scenario, and the datasets used in this study may not fully represent real-world conditions. In summary, the Beyond Orion compression scheme is introduced as an innovative solution for IoT devices, offering superior compression factor and saving percentage compared to established algorithms like Zlib, LZMA, and LZ4. The algorithm demonstrates consistent performance across datasets of different sizes, indicating its potential application in IoT devices. However, it is necessary to address limitations related to compression time and explore opportunities for investigating energy efficiency. These areas present avenues for future research and improvement in this domain.

ACKNOWLEDGEMENT

This work was supported by the Malaysian Ministry of Education Grant FRGS/1/2022/ICT09/SYUC/03/1. The authors would like to thank members of the INTERNAL GRANT SCHEME (IGS) 2023 GRTIN-IGSDCIS-07-2023 for their contribution and collaboration.

This work is part of the development from Sunway University Project STR-RCGS-E-CITIES[S]-004-2021.

REFERENCES

[1] N. Jain, A. Chaudhary, N. Sindhwani, and A. Rana, "Applications of wearable devices in iot," in 2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), pp. 1–4, 2021.

[2] Z. Adiguzel, "Examining strategic fit and innovation in terms of competitive strategies and knowledge management," in Handbook of Research on Strategic Fit and Design in Business Ecosystems, pp. 25–52, IGI Global, 2020.

[3] K. Shafique, B. A. Khawaja, F. Sabir, S. Qazi, and M. Mustaqim, "Internet of things (iot) for next-generation smart systems: A review of current challenges, future trends and prospects for emerging 5g-iot scenarios," IEEE Access, vol. 8, pp. 23022–23040, 2020.

[4] A. Rawat and A. Pandey, "Recent trends in iot : A review," Journal of Management and Service Science (JMSS), vol. 2, p. 1–12, Nov. 2022.

[5] I. Yaqoob, E. Ahmed, I. A. T. Hashem, A. I. A. Ahmed, A. Gani, M. Imran, and M. Guizani, "Internet of things architecture: Recent advances, taxonomy, requirements, and open challenges," IEEE wireless communications, vol. 24, no. 3, pp. 10–16, 2017.

[6] S. Yousefi, H. Karimipour, and F. Derakhshan, "Data aggregation mechanisms on the internet of things: A systematic literature review," Internet of Things, vol. 15, p. 100427, 2021.

[7] The Editors of Encyclopedia Britannica, Morse Code. Jan. 2023.

[8] F. Marcelloni and M. Vecchio, "A simple algorithm for data compression in wireless sensor networks," IEEE Communications Letters, vol. 12, no. 6, pp. 411–413, 2008.

[9] H. M. Al-Kadhim and H. S. Al-Raweshidy, "Energy efficient data compression in cloud based iot," IEEE Sensors Journal, vol. 21, no. 10, pp. 12212– 12219, 2021.

[10] W. Kinsner and R. Greenfield, "The lempel-ziv-welch (lzw) data compression algorithm for packet radio," in [Proceedings] WESCANEX '91, pp. 225–229, 1991.

[11] T. Sheltami, M. Musaddiq, and E. Shakshuki, "Data compression techniques in wireless sensor networks," Future Gener. Comput. Syst., vol. 64, p. 151–162, nov 2016.

[12] B. R. Stojkoska and Z. Nikolovski, "Data compression for energy efficient iot solutions," in 2017 25th Telecommunication Forum (TELFOR), pp. 1–4, 2017.

[13] T. Dang, N. Bulusu, and W.-c. Feng, "Rida: A robust information-driven data compression architecture for irregular wireless sensor networks," in Wireless Sensor Networks (K. Langendoen and T. Voigt, eds.), (Berlin, Heidelberg), pp. 133–149, Springer Berlin Heidelberg, 2007.

[14] A. Ukil, S. Bandyopadhyay, and A. Pal, "Iot data compression: Sensoragnostic approach," in 2015 Data Compression Conference, pp. 303–312, 2015.

[15] M. R. Chowdhury, S. Tripathi, and S. De, "Adaptive multivariate data compression in smart metering internet of things," IEEE Transactions on Industrial Informatics, vol. 17, no. 2, pp. 1287–1297, 2021.

[16] U. Jayasankar, V. Thirumal, and D. Ponnurangam, "A survey on data compression techniques: From the perspective of data quality, coding schemes, data type and applications," Journal of King Saud University – Computer and Information Sciences, vol. 33, no. 2, pp. 119–140, 2021.

[17] I. M. Pu, "Chapter 1 - introduction," in Fundamental Data Compression (I. M. Pu, ed.), pp. 1–17, Oxford: Butterworth-Heinemann, 2006.

[18] A. Hanumanthaiah, A. Gopinath, C. Arun, B. Hariharan, and R. Murugan, "Comparison of lossless data compression techniques in low-cost low-power (lclp) iot systems," in 2019 9th International Symposium on Embedded Computing and System Design (ISED), pp. 1–5, 2019.

[19] A. Vallimayil, K. M. K. Raghunath, V. R. S. Dhulipala, and R. M. Chandrasekaran, "Role of relay node in wireless sensor network: A survey," in 2011 3rd International Conference on Electronics Computer Technology, vol. 5, pp. 160–167, 2011.

[20] J. B. Ernst, "Chapter 14 - energy-efficient next-generation wireless communications," in Handbook of Green Information and Communication Systems (M. S. Obaidat, A. Anpalagan, and I. Woungang, eds.), pp. 371–392, Academic Press, 2013.

[21] H. A. AL-Khafaji, H. M. AlSabbagh, A. Al-Omary, and H. Al-Rizzo, "Influence of relays location and propagation environment on the ber of multiple relay systems," KnE Engineering, vol. 3, p. 314–324, Oct. 2018.

.