IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 7, 2018, accepted September 13, 2018, date of publication October 1, 2018, date of current version October 29, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2872778

Differential-Clustering Compression Algorithm for Real-Time Aerospace Telemetry Data

XUESEN SHI[1], YUYAO SHEN[2], YONGQING WANG[1], (Member, IEEE), AND LI BAI[3],

[1]School of Information and Electronics, Beijing Institute of Technology, Beijing 100081, China
[2]Academy of Opto-Electronics, Chinese Academy of Sciences, Beijing 100094, China
[3]Beijing Aerospace Automatic Control Institute, China Academy of Launch Vehicle Technology, Beijing 100070, China

Corresponding author: Yuyao Shen (syyxyz@gmail.com)

This work was supported by the National Postdoctoral Program for Innovative Talents under Grant BX201700246.

ABSTRACT The volume of telemetry data is gradually increasing, both because of the increasingly larger number of parameters involved and the use of higher sampling frequencies. Efficient data compression schemes are therefore needed in space telemetry systems to improve transmission efficiency and reduce the burden of required spacecraft resources, in particular of their transmitter power. In this paper, a differential-clustering (D-CLU) compression algorithm for lossless compression of real-time aerospace telemetry data is proposed. Because of the temporal-spatial correlation characteristics of telemetry data, the use of a differential compression strategy can efficiently improve compression performance. However, differential compression faces two non-negligible problems, reliability and compression ratio, both of which may be solved by clustering. This is the approach pursued in the proposed D-CLU compression algorithm. The algorithm involves both clustering and coding. In the clustering stage, a one-pass clustering method based on a similarity metric is used to group the original data into clusters. In the coding stage, two traditional encoding algorithms, Lempel–Ziv–Welch and run-length encoding, are used to encode the data, based on the clustering results. Compared with the direct use of differential compression, the clustering-based differential compression algorithm can reduce the error propagation range, thus increasing reliability. The experimental results demonstrate that the proposed D-CLU algorithm can also achieve better compression performance than the other existing algorithms.

INDEX TERMS Real-time aerospace telemetry data, lossless compression, similarity metric, clustering.

I. INTRODUCTION

Aerospace telemetry is a process where the internal or external operating parameters of a spacecraft are collected by the sensors of a data acquisition system, and are then transmitted to a ground station through a communication channel. Users are then provided feedback after analysis by the ground station. Aerospace telemetry systems are typically used to monitor the environmental parameters of spacecraft and their subsystems, thus providing a basis for fault analysis and data processing [1], [2]. In space telemetry systems, both the storage memory and the transmission channel bandwidth are limited; data compression is therefore necessary to improve the transmission efficiency and decrease the transmitter power requirements.

Data compression, which includes both lossy and lossless compression, is the process of removing redundant information from the original data. In telemetry systems, the ground station needs to obtain complete and accurate information to estimate the state of the spacecraft. Therefore, lossless compression is a preferable choice. Several lossless compression solutions are currently used for telemetry data. In [3], the author investigated the Huffman and Arithmetic algorithms; however, the use of a single compression algorithm for all types of data usually results in a lower compression ratio (CR). In [4], a compression algorithm combining both the move to front (MTF) and run-length encoding (RLE) approaches was shown to improve the CR compared to the direct use of RLE. The Rice algorithm proposed in [5], which is recommended by the Consultative Committee for Space Data Systems, consists of two separate functional parts: the preprocessor and the adaptive entropy coder; the effectiveness of this algorithm is determined by the prepro-

cessor performance. As can be seen from [4] and [5], some data preprocessing is performed before encoding to improve the CR; unfortunately, the complexity of these preprocessing-based algorithms may be higher. In [6], a simple and fast compression method is proposed, which can be referred to as deleting the unchanged bytes (DUB); however, it may be ineffective when the compared packets differ by half or more than half the bytes; in addition, this technique suffers from an error propagation problem that may decrease the reliability of the telemetry system.

The effectiveness of lossless compression methods is largely determined by the properties of the telemetry data [7], especially the temporal-spatial correlation (also known as similarity). It is known that clustering techniques can be used to find correlations among sensor data [8]. Data clustering partitions a data set into clusters according to certain similarity indices, such that members within a cluster are somehow similar and members of different clusters are dissimilar [9]. By grouping similar data into a cluster, the compression performance can be significantly improved by preprocessing the cluster members. In addition, while the occurrence of errors in one frame may affect the results of decoding subsequent frames, leading to poor system reliability, clustering-based compression guarantees that the errors will only affect the decoding results in the cluster within which they occur, which can reduce the error propagation range and thus enhance reliability.

Considering the aforementioned advantages of clustering-based compression, we propose the differential-clustering (D-CLU) compression algorithm in this paper. This clustering method is based on similarity metric; two traditional encoding algorithms, RLE and Lempel-Ziv-Welch (LZW) are then used to perform compression based on the clustering results.

This paper is organized as follows. After a brief introduction to the telemetry system model and the data characteristics in Section II, Section III describes the proposed clustering method, which is based on a similarity metric, as mentioned above. Section IV then presents the clustering-based compression and decompression processes, and discusses the most important associated parameters. In Section V, the proposed compression algorithm is evaluated on six test datasets, and performance comparisons are made with other algorithms. Section VI concludes the paper.

II. SYSTEM MODEL AND TELEMETRY DATA CHARACTERISTICS

A simple diagram of the data transfer model of real-time telemetry systems is shown in Fig. 1. The telemetry data are received in real time by the various sensors in the telemetry terminals [10]. After data compression, these data are continuously transmitted to the ground station, frame by frame, so that the ground station may obtain the complete and accurate information required to estimate the state of the spacecraft. Generally, the original telemetry data comprise measurements of multiple parameters obtained by multiple sensors, all sampled with the same sampling rate.

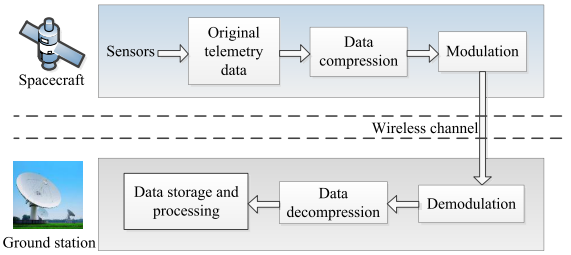


FIGURE 1. Telemetry system data transfer model.

Suppose that $\mathbf{x}_i = \{x_{i1}, x_{i2}, \cdots, x_{ij}, \cdots, x_{in}\}$ represents the sample data collected at time i, where x_{ij} denotes the j-th-th data element. The original data sampled over a certain period of time can then be represented by a telemetry matrix $\boldsymbol{X}_{m\times n}$, as follows:

$$\boldsymbol{X}_{m\times n} = \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_i \\ \vdots \\ \boldsymbol{x}_m \end{bmatrix} = \begin{bmatrix} x_{11} & \cdots & x_{1j} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i1} & \cdots & x_{ij} & \cdots & x_{in} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mj} & \cdots & x_{mn} \end{bmatrix}, \quad 1 \le i \le m, \quad 1 \le j \le n, \quad (1)$$

where m and n are the number of sampling moments and the number of elements collected at each sampling instant, respectively.

The parameters of the telemetry system can be divided into two categories: slowly-varying parameters and fast-varying parameters [11]. For slowly-varying parameters, the observed values change slowly, or remain unchanged for consecutive periods; for fast-varying parameters, the observed values change rapidly or frequently over time. A high sampling rate is therefore needed for fast-varying parameters to avoid information loss; for consistency, the slowly-varying parameters are also sampled at this high sampling rate, which may generate a considerable amount of redundant information. This redundant information causes a high degree of similarity between consecutive sampling times, and therefore leads to high levels of (temporal) correlation between neighboring rows of the telemetry matrix. In addition, the parameter values obtained from neighboring sensor nodes are usually similar, leading to high levels of (spatial) correlation within the rows of the telemetry matrix. Therefore, the telemetry data are correlated in two dimensions: temporal and spatial.

To directly observe the temporal-spatial correlation characteristics of telemetry data, Fig. 2(a) and (b) show two randomly selected row vectors $\boldsymbol{x}_i$ and $\boldsymbol{x}_c$ obtained at two consecutive sampling times. As shown, the data are distributed irregularly in each row vector (i.e. along the different parameters), but the data corresponding to the same parameter are almost identical at the two consecutive sampling times, with only a small fraction of the parameters showing any considerable difference. Fig. 2(c) shows the difference vector

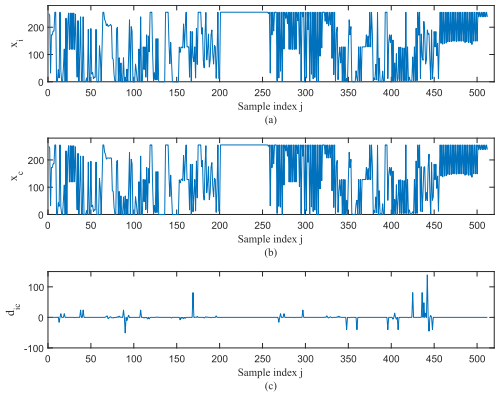


FIGURE 2. Data distribution for x_i, x_c and d_{ic}.

between $\boldsymbol{x}_i$ and $\boldsymbol{x}_c$, which is obtained as

$$\boldsymbol{d}_{ic} = \boldsymbol{x}_i - \boldsymbol{x}_c = \{x_{i1} - x_{c1}, \cdots, x_{in} - x_{cn}\}, \tag{2}$$

where $i = c + \Delta$ $(c, \Delta = 1, \ldots, m-1;\ \ i = 2, \ldots, m)$. Because of the similarity between two consecutive time samples, the difference vector $\boldsymbol{d}_{ic}$ has long continuous sequences of zeros, and it can therefore be efficiently compressed using the RLE algorithm. To the best of our knowledge, RLE is the best choice in this context for real-time systems, given its fast compression and decompression performance [12]. Moreover, as can be seen from Fig. 2(a) and 2(b), the (spatial) correlation existing within single row vector makes it possible to compress the data within those row vectors, a task for which the LZW algorithm is a better choice [13].

According to the above analysis, two-dimensional compression combining differential compression between two vectors with temporal correlation and simple compression within the individual vectors presenting spatial correlation will be more efficient for telemetry data. However, when differential compression is performed on two vectors, two problems will be appeared, which cannot be ignored.

(1) Differential compression is more efficient on similar vectors than dissimilar vectors. If differential compression is used on two dissimilar vectors, the CR will be lower than that obtained by compressing a single vector with a suitable algorithm. Therefore, in order to obtain a satisfactory CR, it is important to distinguish between similar and dissimilar vectors.

(2) The compressed data are susceptible to some degree of error propagation. Errors occurring in one frame during data transmission may affect the decoding results of subsequent frames, which lead to poor system reliability. The DUB algorithm in [4], for example, deletes the unchanged bytes between adjacent packets; all subsequent decoding results are damaged once an error occurs in one packet. Therefore, our compression algorithm should overcome this error propagation problem.

A clustering-based compression strategy may solve the above problems. Clustering is a useful and ubiquitous tool in data analysis. Similar data vectors are assigned to the same cluster of a partition according to a chosen similarity metric, whereas dissimilar vectors are assigned to different clusters. Differential compression performed within a cluster created by such clustering methods may therefore ensure higher CRs. Furthermore, the compressed data vectors obtained from different clusters are independent of each other; therefore, the errors occurring in one frame will not affect the decoding results in other clusters.

We therefore propose here a clustering-based compression algorithm. This algorithm contains two main components: clustering and coding. The details of the clustering method are presented in Section III, and the compression and decompression procedures are described in Section IV.

III. CLUSTERING METHOD

In this section, we provide a complete description of the proposed clustering method used to improve compression performance. In Section III.A, we introduce a novel similarity metric, which is then applied to the proposed clustering method. In Section III.B, we propose a data stream clustering method that can meet the requirements of real-time compression and transmission of telemetry data.

A. SIMILARITY METRIC

The choice of a similarity metric is a critical step in clustering. The similarity metric is used to indicate how suitable a pair of vectors is for inclusion in the same cluster [14]. A large similarity value implies that those vectors are highly likely to belong to the same cluster. Therefore, the effectiveness of clustering algorithms always depends on the appropriateness of the similarity measures to the data being processed. A similarity measure proposed in [15] is based on considering that two objects are more similar when one can be further compressed given the information in the other. This means that the CR can be used as a similarity measure index. Based on this inspiration, we present here a novel compression-based similarity measure between data vectors.

Our first objective is to derive an appropriate model for measuring similarity between data vectors. As discussed in Section II, the difference vector would be more efficiently compressed by RLE than by the other traditional compression algorithms. Therefore, we designed the proposed similarity measure based on the RLE algorithm. The CR of RLE has a strong dependence on the redundancy distribution characteristics (RDC) of the data. Therefore, the similarity may be obtained by analyzing the RDC of the difference vector.

In order to analyze the RDC of the elements in a given vector, a variable Y is defined, which denotes the number of repetitions of the same byte value in consecutive locations; the possible range of variable Y is $y_k \in \{1, 2, 3, 4, \cdots, n\}$, and its probability distribution is $P\{Y = y_k\} = p_k$. The mathematical expectation $E(Y)|_{\boldsymbol{u}}$ is used as an indicator of the RDC of vector $\boldsymbol{u}$, and can be calculated as

$$E(Y)|_{\boldsymbol{u}} = \sum_{k=1}^{n} p_k y_k = \frac{n}{N}, \tag{3}$$

TABLE 1. The probability distribution of Y.

Y	1	2	3	4	5	6	...	16
p_k	3/7	2/7	0	1/7	1/7	0	...	0

where $N = \sum_{k=1}^{n} N_k$, and N_k is the number of y_k occurrences. An example is now given to illustrate the calculation of $E(Y)|_{\boldsymbol{u}}$. Let us consider a given vector $\mathbf{u} = \{u_1, u_1, u_2, u_1, u_1, u_1, u_1, u_3, u_3, u_2, u_1, u_1, u_1, u_1, u_1, u_2\}$, for which $n = 16$ and $N = 7$; the probability distribution of Y can be estimated as shown in Table 1.

From (3), the mathematical expectation is $E(Y)|_{\mathbf{u}} = 1 \times \frac{3}{7} + 2 \times \frac{2}{7} + 4 \times \frac{1}{7} + 5 \times \frac{1}{7} = \frac{16}{7}$. To obtain the relationship between CR and $E(Y)|_{\boldsymbol{u}}$, we use RLE to compress the vector $\boldsymbol{u}$; and calculate CR as follows:

$$CR = (1 - \frac{z'}{z}) \times 100\% = (1 - \frac{2N}{n}) \times 100\%, \tag{4}$$

where z and z' denote the original data size and the compressed data size in bytes, respectively. From (3) and (4), we obtain the relationship between CR and $E(Y)|_{\boldsymbol{u}}$ as

$$CR = (1 - \frac{2}{E(Y)|_{\boldsymbol{u}}}) \times 100\%. \tag{5}$$

It can be seen from (5) that the greater the mathematical expectation value is, the higher the CR becomes. Based on this analysis, we define the similarity measure between two vectors as

$$S(\boldsymbol{x}_i, \boldsymbol{x}_c) = E(Y)\big|_{\boldsymbol{d}_{ic}}, \tag{6}$$

where $\boldsymbol{d}_{ic}$ is the difference vector between $\boldsymbol{x}_i$ and $\boldsymbol{x}_c$.

B. CLUSTERING

The aerospace telemetry data are real-time data streams [10], continuously and sequentially transmitted, frame by frame, to the ground station. Given that a data stream is an ordered and potentially unbound sequence of data points, the traditional clustering algorithms are difficult to apply [16]. Specific data stream clustering algorithms usually differ from the traditional clustering algorithms in three important characteristics [17], [18]:

(1)The number of clusters is previously unknown. New cluster will emerge as the data points are input to the algorithm.

(2)The data points can be accessed only once or a small number of times in the order of data inflow.

(3)The algorithm should have the ability to analyze and process outliers.

Data stream clustering is a process where new clusters emerge unceasingly, while old clusters disappear, as all data points are sequentially read to form clusters [17]. Most data stream clustering algorithms proposed in recent years contain two steps [17], [19], [20]: an online and an offline step. During the online step, micro-clusters are created, using a single pass over the data; these micro-clusters are then re-clustered into macro-clusters in the offline step. Although these existing data stream clustering algorithms can efficiently group data into clusters with high quality, they are not directly applicable to our real-time data compression and transmission context. To satisfy the real-time requirement, we will only consider a one-pass clustering strategy, i.e., one in which all data are clustered in a single step, without re-clustering. Furthermore, the outliers in our method are compressed and transmitted, instead of being deleted as in other algorithms.

The proposed clustering method is described below. It is inspired on the idea of micro-cluster creation present in the online step of data stream clustering. A cluster contains a set of similar data points with a cluster head (CH) (also known as cluster center) and several cluster members (CMs), and is created using a similarity metric, in real time, as the data points arrive.

Our clustering method considers each row of the telemetry matrix as a data point in the data streams, and each data point as a potential CH (P-CH) (the concept of P-CH is first defined in [21]). Let $C_{\boldsymbol{x}_c}$ denote the cluster currently being processed whose real CH is $\boldsymbol{x}_c$, and let $\boldsymbol{x}_c^p$ denote the current P-CH. A CH-buffer is defined, which can store either a real CH or a current P-CH. The existence of a real CH in the CH-buffer implies that the current cluster exists; otherwise, there is no current cluster. In addition, we define parameter k_p to represent the cluster width (the number of elements in the current cluster $C_{\boldsymbol{x}_c}$); its upper limit value is K. The cluster width has a great influence on the compression performance; this influence depends on the design of the compression process. Therefore, we will provide a detailed discussion of the cluster width in Section IV.C, and on how to obtain the optimal value of K in Section V. The similarity threshold V_h(to be discussed below) is another important parameter to determine the cluster width k_p in view of similarity, and will be discussed in Section V.

The overall proposed clustering method is summarized in Algorithm 1. Once a new data point is input, the clustering process is enabled. At the beginning, the similarity between the new input data point $\boldsymbol{x}_i$ and the data point in the CH-buffer is calculated, effectively starting the clustering process. Two different processes flow are followed within the clustering procedure:

Case one: the data point in the CH-buffer is a current P-CH, $\boldsymbol{x}_c^p$. If the similarity between $\boldsymbol{x}_i$ and $\boldsymbol{x}_c^p$ exceeds the threshold V_h, a new current cluster is created and this P-CH is upgraded to a real CH. Otherwise, $\boldsymbol{x}_c^p$ is considered to be an outlier and removed from the CH-buffer. Then, $\boldsymbol{x}_i$ is loaded into the CH-buffer as the new current P-CH.

Case two: the data point in the CH-buffer is the current real CH, $\boldsymbol{x}_c$, of $C_{\boldsymbol{x}_c}$. If k_p is below K, and the similarity between $\boldsymbol{x}_i$ and $\boldsymbol{x}_c$ exceeds the threshold, this new input data point $\boldsymbol{x}_i$ is assigned to $C_{\boldsymbol{x}_c}$. Otherwise, the current cluster process is terminated; then, $\boldsymbol{x}_c$ is removed from the CH-buffer and $\boldsymbol{x}_i$ is loaded into the CH-buffer as the new current P-CH.

Algorithm 1 Clustering Method

Input: the new data point $\boldsymbol{x}_i \in \boldsymbol{X}_{m\times n}$
  if the data point in the CH-buffer is $\boldsymbol{x}_c^p$ **then**
    Calculate $S(\boldsymbol{x}_i, \boldsymbol{x}_c^p)$
    if $S(\boldsymbol{x}_i, \boldsymbol{x}_c^p) \geq V_h$ **then**
      $\boldsymbol{x}_c^p$ is upgraded to $\mathbf{x}_c$, and a new $C_{\boldsymbol{x}_c}$ is created.
    else
      Remove $\boldsymbol{x}_c^p$ from the CH-buffer and load $\boldsymbol{x}_i$ into the CH-buffer as the new current P-CH.
    end if
  end if
  if the data point in the CH-buffer is $\boldsymbol{x}_c$ **then**
    Calculate $S(\boldsymbol{x}_i, \boldsymbol{x}_c)$
    if $S(\boldsymbol{x}_i, \boldsymbol{x}_c) \geq V_h$ and $k_p < K$ **then**
      Merge $\boldsymbol{x}_i$ into the current cluster $C_{\boldsymbol{x}_c}$;
    else
      Remove $\boldsymbol{x}_c$ from the CH-buffer and load $\boldsymbol{x}_i$ into the CH-buffer as the new current P-CH.
    end if
  end if

IV. COMPRESSION AND DECOMPRESSION

In this section, we provide a brief description of the clustering-based compression and decompression processes. In addition, we present a detailed discussion of the cluster width parameter.

A. COMPRESSION

In general, the selection of a suitable compression algorithm depends on the correlation characteristics of the data so that better compression performance can be achieved. In the proposed algorithm, two traditional compression algorithms are combined to encode the original telemetry data, considering the two types of correlation present in those data. As described in Section II, if we consider both the correlation characteristics and the real-time requirement, the RLE algorithm becomes a better choice to compress the difference vector obtained from two similar vectors. The clustering process ensures that all members in a cluster are similar to their CH. Therefore, we use RLE to encode the difference vector between each CM and its CH. Moreover, CHs and outliers should be encoded to further enhance the overall CR of the telemetry data. The LZW algorithm is used to encode them. We note that all CHs and outliers originate in P-CHs; therefore, in order to avoid repeated encoding, once a P-CH is determined, it is compressed with LZW, instead of performing the compression on a real CH or outlier.

The compression algorithm accepts each row vector of the telemetry matrix as an input at each time period. The main steps of the compression procedure are

(1) Take $\boldsymbol{x}_1$ as the first P-CH, and load it into the CH-buffer. Then, encode it with LZW and transmit it.

(2) As the next vector is input, the clustering process starts, as described in Section III.B. If this new input vector is classified as a new P-CH, encode it with LZW and transmit it; otherwise, encode the difference vector between the new input and the vector in the CH-buffer with RLE, and transmit it. Repeat this step as long as there are new input vectors.

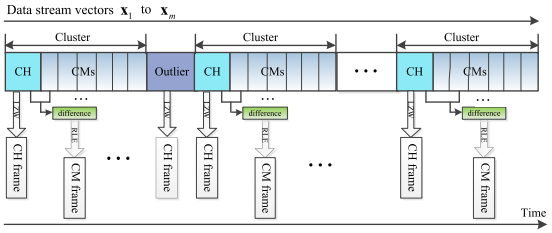


FIGURE 3. Clustering-based real-time compression and transmission.

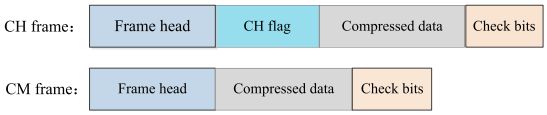


FIGURE 4. Frame format.

The most important characteristic of the proposed algorithm is the real-time compression and transmission. To achieve it, we perform data clustering and encoding simultaneously. Once a new input is classified as a P-CH or a CM, the encoding procedure is enabled, according to the encoding strategy described above. The compressed data are organized into a frame format before being transmitted. Fig. 3 shows a simple diagram of the clustering-based real-time compression and transmission procedure. For convenience, a transmission frame containing a compressed CH or outlier referred to as a CH frame; otherwise, it is referred to as a CM frame. A CH flag is added to each CH frame in order to identify CH frames and allow a correct data decompression. Check bits are added to all frames, thus implementing an error detecting code. The transmission frame format is shown in Fig. 4.

B. DECOMPRESSION

D-CLU is a lossless compression algorithm, and therefore enables a full reconstruction of the original data at the ground station through the decompression procedure. This procedure follow the inverse process flow used in the compression step: it uses LZW to decode the CH frames, and RLE to decode the CM frames.

A CH-buffer is also defined at the receiving end to store a CH frame, which is then used to recover the other CMs in the corresponding cluster. Once a CH frame is detected by identifying a CH flag, it is decoded, and the decoded data are loaded into the CH-buffer; the processing then proceeds to the next frame. Depending on the type of frame immediately following a CH frame, two different procedures are followed:

Case one: another CH frame is detected. Remove the former CH frame from the CH-buffer, and load the currently decoded CH frame into the CH-buffer.

Case two: a CM frame is detected. Decode it using the RLE algorithm, and then add the decoded data to the data stored in the CH-frame, thus retrieving the original data.

Error control is an important issue in lossless compression because the decompressed data will be damaged if errors occur during transmission in noisy channels. In our algorithm, check bits are added in the transmission frames for error detection. When an error is detected in a received frame, two recovery procedures are executed:

(1) If the frame in error is a CH frame, all the data in the current cluster are discarded. This is a direct consequence of the fact that all the CMs in a cluster are recovered using their CH, and therefore, an error in the CH frame will damage all the CMs of that cluster.

(2) If the frame in error is a CM frame, discard only this frame. Given that an error in a CM frame cannot affect the other CMs in the cluster, they can still be correctly recovered using their CH.

From the above discussion, we can see that a frame error can only affect the decoded results belonging to the same cluster, and will not, in any way, be propagated to other clusters. Therefore, our proposed algorithm can significantly enhance the reliability performance when compared with the direct use of differential compression. Moreover, we note that there is no clustering analysis in the decompression process; therefore, the decompression time (DT) of the proposed algorithm will theoretically be shorter than the compression time (CT).

C. CLUSTER WIDTH

The cluster width, k_p, was defined in Section III.B as being the number of vectors in the current cluster. This subsection discusses this parameter in detail.

The selection of a cluster width value has a significant influence on the reliability and compression performance of the proposed algorithm. On the one hand, cluster width affects reliability. The smaller the value of k_p, the smaller the error propagation range on the decompressed data, given that error propagation is always limited to the cluster where the error occurred. Therefore, having too many members per cluster may lead to poor system reliability. Moreover, cluster width affects the compression performance. Intra-cluster compression, in the proposed algorithm, combines LZW and RLE algorithms to process CH vectors and difference vectors. A different value of k_p implies a different number of vectors to be compressed by each one of these compression algorithms, which leads to different compression performances. With increasing cluster width, the number of vectors that need to be compressed by LZW will be reduced, resulting in higher CRs and shorter CTs. Therefore, from the point of view of reliability, the value of k_p should be as smalle as possible; the opposite is true in terms of compression performance. To balance these two aspects, we define another parameter K, which corresponds to the upper limit of k_p. The optimal value of K will be experimentally determined in Section V.

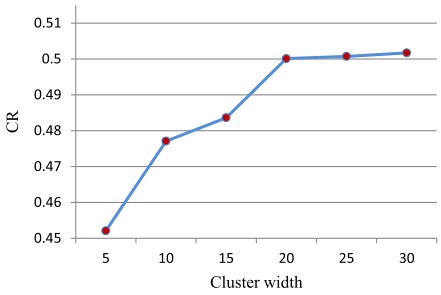


FIGURE 5. CR for different values of cluster width.

V. PERFORMANCE EVALUATION

In this section, the performance of the proposed compression algorithm is evaluated through numerical simulation using the MATLAB software package. First, we experimentally determine the optimal values of two of the methodąŕs important parameters. Then, we evaluate the compression performance of the proposed algorithm in terms of CR, CT and DT, and compare this performance with those of several other existing algorithms.

The evaluation is performed using six real-world datasets from a certain aerospace application. Each dataset is processed into a telemetry matrix, as described in Section II. The resulting telemetry data matrices contain 2000 rows of 512 bytes, for a total of 1024 KB. The uncompressed original telemetry data contain multiple parameter samples and additional service information. All the original and differential data are presented in 8-bit complementary code.

A. PARAMETERS

In the proposed method, we need to experimentally find the optimal value of two parameters: the upper limit of the cluster width, K and the similarity threshold V_h. Both these parameters have a significant influence on the compression performance.

1) CLUSTER WIDTH

The choice of K should consider the need to balance the relationship between compression performance and the error propagation effect. According to the analysis in Section IV.C, an optimal value of K should be found at the minimum value of k_p for which satisfying compression performances are obtained. We therefore simulate the CR, CT, and DT under different values of cluster width, and find an optimal value of K that guarantees the desired compression performance. The compression performance obtained with the D-CLU algorithm for different values of k_p is shown in Fig. 5 and 6. As shown, both CR and CT improve monotonically when the value of k_p increases from 5 to 30; these improvements become less significant when the value of k_p exceeds 20. This effect can be understood by noticing that, for data vectors with high levels of similarity, the CR of LZW applied to a single vector is lower than that of RLE applied to a difference vector; additionally, the compression delay of

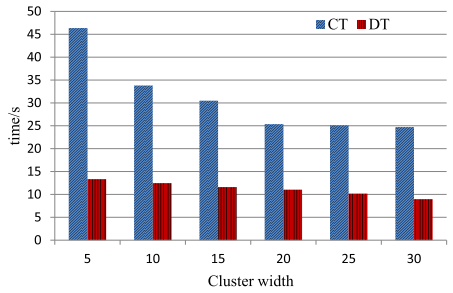


FIGURE 6. CT and DT for different values of cluster width.

TABLE 2. CRs obtained using different single algorithms (%).

Algorithms	D-1	D-2	D-3	D-4	D-5	D-6	Average
RLE	20.57	29.52	28.04	28.98	19.24	31.22	26.26
LZW	28.48	32.43	29.52	41.24	30.12	35.28	32.84
Huffman	30.28	29.67	30.12	34.23	37.61	35.12	32.83
arithmetic	31.45	31.29	30.57	32.83	35.25	30.26	31.94
DUB[4]	33.32	31.02	29.01	42.46	30.56	30.06	32.74

LZW is larger than that of RLE. Therefore, the compression performance is fundamentally affected by the number of vectors compressed using LZW. With increasing cluster width, the influence of this parameter on compression performance becomes smaller because the proportion of CHs requiring LZW compression in the whole telemetry matrix is gradually reduced; when the cluster width exceeds 20, its influence on the compression performance decreases. Because the decompression delay of LZW is basically the same as that of RLE, the cluster width has only a small effect on DT. Therefore, the optimal value of K selected for the proposed method is 20.

2) SIMILARITY THRESHOLD

The similarity threshold, V_h, is another important parameter in the proposed algorithm. A higher threshold implies smaller cluster width, which may lead to lower CRs and longer CTs; a lower threshold, on the contrary, may shorten the CT, but may also reduce the CR, given that relatively dissimilar vectors will be assigned to the same cluster. In this paper we will experimentally determine the optimal value of V_h. The proposed algorithm should ensure that the CR of differential compression with RLE is higher than that of single vector compression. Therefore, we simulate the CRs obtained using different single compression algorithms for all the original vectors; the obtained results are shown in Table 2. The average CR is 31.32%, and the highest CR of an individual algorithm over all sets is 32.84%. The relationship between similarity and CR is shown in Fig. 7. Clearly, the CR improves with the increase of similarity; when $S(\mathbf{x}_i, \mathbf{x}_c) \geq 3$, $CR > 33\%$ is expected. Based on these experimental results, we chose a value of $V_h = 3$ for our simulation.

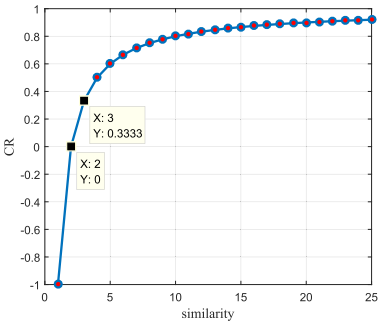


FIGURE 7. Relationship between CR and similarity.

TABLE 3. CRs of different lossless algorithms (%).

Algorithms	D-1	D-2	D-3	D-4	D-5	D-6	Average
MTF+RLE[2]	35.32	38.21	39.57	40.19	36.73	37.76	37.96
Rice[3]	48.59	46.76	48.21	50.45	47.89	48.23	48.36
D-CLU	**50.21**	**51.34**	**49.05**	**50.98**	**48.37**	**51.54**	**50.24**

B. COMPRESSION PERFORMANCE

The compression performance is usually evaluated by considering the CR, CT, and DT. In this subsection, we will follow this approach and evaluate the proposed algorithm. Comparisons will also be made with other existing algorithms.

1) COMPRESSION RATIO

As shown in Table 2, applying a single compression algorithm (e.g., RLE, LZW, Huffman, Arithmetic and DUB) to the original data usually leads to a lower CR. To evaluate the compression performance of the proposed algorithm and compare it with those of other preprocessing algorithms, a simulation was performed and is presented in this subsection. Table 3 shows the CRs for the six datasets (D-1 to D-6) using different lossless compression algorithms. The best CR for each dataset is highlighted in bold. The last column in Tables 3 shows the average values of the CRs in the previous columns. We observe that appropriately preprocessing the data before coding (e.g., MTF+RLE, Rice and D-CLU) can improve the CR. Furthermore, the average CR of the proposed algorithm is better than those of the other algorithms.

2) COMPRESSION AND DECOMPRESSION TIMES

The compression and decompression speeds are also important attributes when measuring the performance of compression algorithms. Therefore, we evaluated the proposed algorithm an evaluation in terms of CT and DT on the six datasets (D-1 to D-6). Only preprocessing-based compression algorithms are considered in this subsection, because the traditional single compression algorithms always obtain lower values of CR. The CT and DT of the algorithms are obtained using a timing function. The simulation results are shown in Fig. 8 and 9; shorter CTs and DTs imply better real-time performances. As shown, both the CT and

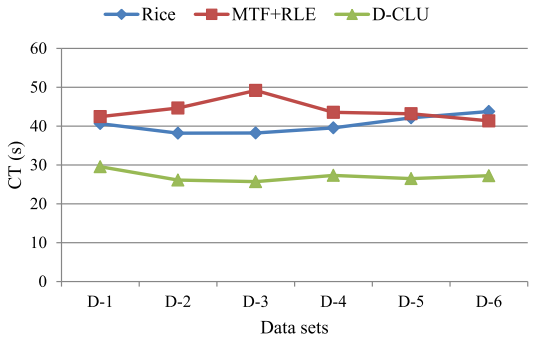


FIGURE 8. CTs of the different algorithms.

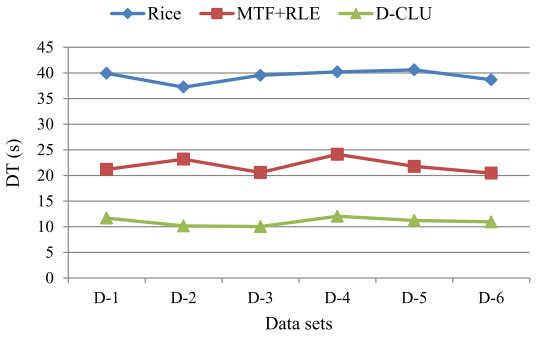


FIGURE 9. DTs of the different algorithms.

DT performances of D-CLU are better than those of the other compared algorithms. Moreover, it should be noted that the DTs of the Rice algorithm are similar to its CTs, whereas the MTF+RLE and D-CLU require less time to decompress (DTs) than to compress (CTs). The D-CLU algorithm recovers the original data without similarity calculations, leading to DTs that are shorter than the CTs. In summary, the test results demonstrate that the D-CLU algorithm is faster than both the MTF+RLE and Rice algorithms.

Generally, preprocessing combined with coding will raise the complexity of the compression algorithm. Therefore, the complexity of D-CLU algorithm is higher than that of a single compression algorithm. However, the complexity of D-CLU algorithm is still lower than other preprocessing compression algorithms. The D-CLU algorithm is designed based on data streaming clusters. The original data is clustered and compressed in real-time. This strategy makes it possible to reuse the resources with new data as the input. Therefore, the complexity of the proposed algorithm is lower than other preprocessing algorithms, especially as the size of the data is increased.

VI. CONCLUSION

In this paper, an efficient D-CLU lossless compression algorithm was proposed for real-time aerospace telemetry data. A clustering method was designed based on a proposed similarity metric. Because of the temporal-spatial correlation of the telemetry matrix, the RLE method was combined with LZW encoding for inter-row and intra-row compression of the vectors in each cluster, respectively. It has been experimentally demonstrated that the proposed algorithm can provide better compression performance than the other existing algorithms. Furthermore, the proposed D-CLU algorithm offers some additional advantages. The proposed clustering-based compression algorithm ensures that the compression/decompression of vectors in different clusters are independent processes. Error propagation is therefore contained within the affected cluster, thus enhancing the reliability of the overall system. Even though the proposed algorithm has been designed for telemetry data, it is also suitable for other sensor data with high temporal-spatial correlation characteristics. Moreover, the proposed algorithm can comply with different application requirements by flexibly adjusting its parameter values.

REFERENCES

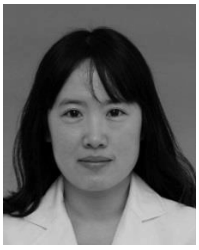
[1] A. Giannini *et al.*, "The sardinia radio telescope upgrade to telemetry, tracking and command: Beam squint and electromagnetic compatibility design," *IEEE Antennas Propag. Mag.*, vol. 57, no. 1, pp. 177–191, Feb. 2015.
[2] Q. Wang, B. Wang, and B. Wu, "Study on threats to security of space TT&C systems," in *Proc. 26th Conf. Spacecraft TT&C Technol. China*, vol. 187, 2013, pp. 67–73.
[3] G. Beglaryan, "Lossless compression of satellite telemetry data for a narrow-band downlink," Ph.D. dissertation, California State Univ., Northridge, Los Angeles, CA, USA, Jun. 2014.
[4] J. G. Abraham, R. Mishra, and J. Deepa, "A lossless compression algorithm for vibration data of space systems," in *Proc. Int. Conf. Next Gener. Intell. Syst. (ICNGIS)*, Feb. 2016, pp. 1–7.
[5] *Lossless Data Compression. Recommendation for Space Data Systems*, Standards CCSDS 121.0-B-2, Blue Book, CCSDS, Washington, DC, USA, May 2012.
[6] L. Guojun, S. Jian, and Z. Running, "Lossless data compression algorithm for satellite packet telemetry data," in *Proc. IEEE Int. Conf. MEC*, Dec. 2013, pp. 2756–2759.
[7] M. Elshafey and I. Sidyakin, "Lossless compression of telemetry information using adaptive linear prediction," *Sci. Educ. Bauman MSTU*, vol. 14, no. 4, pp. 354–363, Apr. 2014.
[8] Y. Yin, F. Liu, X. Zhou, and Q. Li, "An efficient data compression model based on spatial clustering and principal component analysis in wireless sensor networks," *Sensors*, vol. 15, no. 8, pp. 19443–19465, Aug. 2015.
[9] L. Chen, L.-J. Zou, and L. Tu, "A clustering algorithm for multiple data streams based on spectral component similarity," *Inf. Sci.*, vol. 183, no. 1, pp. 35–47, 2012.
[10] X. Dong and D. Pi, "An effective method for mining quantitative association rules with clustering partition in satellite telemetry data," in *Proc. IEEE 2nd Int. Conf. Adv. Cloud Big Data*, Nov. 2015, pp. 26–33.
[11] Y. Ren, X. Liu, W. Xu, and W. Zhang, "Multi-channel data structure and real-time compression algorithm research," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 24, no. 11, pp. 28–35, Nov. 2009.
[12] T. Nishitsuji, T. Shimobaba, T. Kakue, and T. Ito, "Fast calculation of computer-generated hologram using run-length encoding based recurrence relation," *Opt. Express*, vol. 23, no. 8, pp. 9852–9857, 2015.
[13] T. A. Welch, "A technique for high-performance data compression," *Computer*, vol. 17, no. 6, pp. 8–19, Jun. 1984.
[14] C. Surianarayanan and G. Ganapathy, "An approach to computation of similarity, inter-cluster distance and selection of threshold for service discovery using clusters," *IEEE Trans. Services Comput.*, vol. 9, no. 4, pp. 524–536, Jul./Aug. 2016.
[15] M. Li, X. Chen, X. Li, B. Ma, and P. M. B. Vitányi, "The similarity metric," *IEEE Trans. Inf. Theory*, vol. 50, no. 12, pp. 3250–3264, Dec. 2004.
[16] X. Zhang, C. Furtlehner, C. Germain-Renaud, and M. Sebag, "Data stream clustering with affinity propagation," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 7, pp. 1644–1656, Jul. 2014.
[17] F. Cao, M. Ester, W. Qian, and A. Zhou, "Density-based clustering over an evolving data stream with noise," in *Proc. SDM*, 2006, pp. 326–337.

[18] X. Zhang and W. Zeng, "Research and advances of real-time data stream clustering," (in Chinese), *Comput. Eng. Des.*, vol. 30, no. 9, pp. 2177–2181, May 2009.
[19] J. A. Silva, E. R. Faria, R. C. Barros, E. R. Hruschka, A. C. P. L. F. de Carvalho, and J. Gama, "Data stream clustering: A survey," *ACM Comput. Surv.*, vol. 46, no. 1, pp. 1–31, 2013.
[20] M. Hahsler and M. Bolaños, "Clustering data streams based on shared density between micro-clusters," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 6, pp. 1449–1461, Jun. 2016.
[21] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, no. 5814, pp. 972–976, Feb. 2007.

XUESEN SHI was born in Changchun, Jilin, China, in 1984. She received the B.Eng. degree in electronic and information engineering from Yanshan University, China, in 2008. She is currently pursuing the Ph.D. degree with the School of Information and Electronics, Beijing Institute of Technology, Beijing, China. From 2008 to 2014, she was an Engineer with Kingdee and Lenovo Inc. Her research interests include spread spectrum signal processing, spaceflight TT&C, and satellite navigation and positioning.

YUYAO SHEN was born in Handan, Hebei, China, in 1988. She received the B.Eng. and Ph.D. degrees from the Beijing Institute of Technology, Beijing, China, in 2011 and 2017, respectively. She currently holds a post-doctoral position at the Academy of Opto-Electronics, Chinese Academy of Sciences. Her research interests include space communication and spaceflight TT&C, especially spread spectrum signal processing.

YONGQING WANG was born in Bengbu, Anhui, China, in 1981. He received the master's and Ph.D. degrees from the School of Information and Electronics, Beijing Institute of Technology. He is currently pursuing the Ph.D. degree with the Beijing Institute of Technology. From 2003 to 2008, he was with the School of Information and Electronics, Beijing Institute of Technology. He is currently an Associate Professor with the Beijing Institute of Technology. He has done many projects which were funded by the National High Technology Research and Development Program of China. His research interests include spaceflight TT&C, telecommunication technology, electronic systems simulation and signal analog, spread spectrum signal processing theory, and satellite navigation and positioning.



LI BAI was born in Datong, Shanxi, China, in 1980. She received the B.Eng. degree from the Nanjing University of Aeronautics and Astronautics, China, in 2003, and the M.A. degree in mechanical electronics from the Beijing Information Science and Technology University, China, in 2011. From 2011 to 2017, she was an Assistant Researcher with the National Space Center, Chinese Academy of Sciences. Then, she was an Engineer with the Beijing Aerospace Automatic Control Institute, where she was involved in software engineering.