Swinging Door Trending Compression Algorithm for IoT Environments

Juan David Arias Correa
Post-graduate Program of Automation Systems Engineering
Federal University of Santa Catarina (UFSC)
Florianópolis, Brazil – juan.arias@posgrad.ufsc.br

Alex Sandro Roschildt Pinto
Dept. of Informatics and Statistic
Federal University of Santa Catarina (UFSC)
Florianópolis, Brazil – a.r.pinto@ufsc.br

Carlos Montez
Dept. of Automation and Systems Engineering
Federal University of Santa Catarina (UFSC)
Florianópolis, Brazil – carlos.montez@ufsc.br

Erico Meneses Leão
Dept. of Computation
Federal University of Piauí (UFPI)
Teresina, Brazil – ericoleao@ufpi.edu.br

***Abstract*—The transmission and storage of data collected by devices are essential components of the Internet of Things (IoT). When devices transmit irrelevant or redundant sensed data, it spends more energy, unnecessarily using the communication channel, and processing data that make a small contribution to the application. Data compression is a possible solution for the significant amount of data generated by IoT devices by reducing the volume of data necessary to represent the information. This paper proposes the use of Swinging Door Trending (SDT) in IoT applications and a new self-configuration step to select its major parameter: the compression deviation. A prototype was built, and experimental results show the effectivity of the proposal.**

***Index Terms*—Data compression, IoT, Internet of Things.**

I. Introduction

As the area of information technology proliferates, data generation grows faster than the capacity of storage and transmission technologies. The Internet of Thing (IoT) is a trend that has a relevant effect on this growth, due to its paradigm that interconnects things and devices of the real world with the digital world through the Internet. This paradigm makes transparent the separation between the physical and digital world, favoring the creation of new services [1].

Applications can be built combining these services and devices, and this combination gives rise new kind of applications, involving different domains, such as health care, precise agriculture and smart city. An important technology that evolved together with IoT was cloud computing [2].

Cloud computing is a model that allows the on-demand access of a shared set of configurable computing resources such as servers, applications, and services. It can be rapidly launched and provisioned with a minimal management effort and brings significant advantages for IoT. However, the integration of cloud computing into IoT applications imposes new challenges due to the massive amount of data coming from IoT sources [3]. The use of cloud platforms has economic and operational costs demanded by the transportation of information from the device to the cloud, and the storage of IoT data generate additional costs since data continually increase over time.

IoT devices commonly have storing, processing and energy limitation being data transmission a significant source of energy consumption [4], [5]. In general, reducing both the transmitted and stored data are essential from the perspective of IoT and cloud computing, and a possible solution to deal with this question is through data compression techniques [6].

The Swing Door Trending (SDT) is an online and lossy data compression algorithm traditionally used in Supervisory Control and Data Acquisition (SCADA) systems aiming to store historical data from process information systems (PIMs). SDT has low computational complexity and uses a linear trend to represent a quantity of data. Its most important parameter is the compression deviation (CD) that represents the maximum difference between the current sample and the current linear trend used to represent the data previously collected [7].

This paper proposes the use of SDT on IoT devices as a mechanism to reduce the number of data sent and stored on the cloud. The SDT was selected for its low computational complexity and small memory demand, allowing it to be embedded into IoT devices. However, SDT has a well-known disadvantage because the CD parameter should be pre-runtime defined.

The contribution of this paper is to propose mechanisms that allow IoT devices to properly self-determine the value of the CD used locally for compressing its data amount. In this sense, a compression criterion was proposed as a metric to evaluate lossy compression algorithms. This metric takes account the compression error and rate metrics in order to give to the SDT algorithm an adequate balanced metric to represent the quality of the compression.

This paper is divided into six parts. Section 2 presents background about data compression and SDT algorithm. Section 3 presents the proposal. Section 4 shows the related work. Section 5 presents the experimental assessment of the proposal. Section 6 contains final remarks.

This research was supported by CAPES Print Program grant 698503P and FAPEPI/MCT/CNPq/CT-INFRA No 007/2018.

II. Background

A. Data Compression

The demand for data compression is not recent. One of its first techniques was the Morse code in the nineteenth century. Today, the field of information technology is evolving rapidly, resulting in the generation of a massive amount of data, leading to an ever-increasing in data storage and transmission demands. The rate of growth of data is much higher than the rate of growth of storage and transmission technologies [1].

Data compression techniques are a possible solution for dealing with this kind of problem, eliminating redundant data or representing it in a reduced way. These techniques are fundamental in many real-time applications such as satellite imagery, wireless sensor networks (WSN), IoT, and others [1].

Data compression is the process of representing data in a compressed form, optimizing the resource usage [8]. There are two types of data compression algorithms: lossless and lossy. Lossless algorithms can reconstruct the original message exactly from the compressed message. These algorithms are normally used for data that cannot be modified and must be equal than the original data. Lossy algorithms can only reconstruct an approximation of the original messages. They are commonly used in situations that are possible to eliminate unnecessary data, like multimedia and sensing systems that usually presents a lot of redundant data [1].

B. Swinging Door Trending

Swinging Door Trending (SDT) is a data compression method that uses a linear trend to represent a number of samples. It is a lossy compression algorithm and has, as the most important parameter, the Error Compression Deviation (CD) that corresponds to the maximal difference that a point could have to be represented as part of some linear trend [7].

From a 2D representation where the x-axis represents the time and the y-axis the data values to be compressed, SDT creates a parallelogram that has a longitudinal trend line in the center with a starting point at the first value (FV). Figure 1 illustrates an example of five samples. Two limit lines delimit the parallelogram, the upper boundary (UB) and lower boundary (LB), that are sited at a distance CD from the trend line, and that begin, respectively, at the upper pivot (UP) and lower pivot (LP) points [9].

SDT uses sloping lines as reference points, having a sloping upper (SB) and sloping lower (SL) lines beginning in UP in LP, respectively. SB only can move to counter-clockwise (increase) and the SL to clockwise (decrease). When a new value arrives, these slopes may vary to insert it in the coverage area. This change is possible when SB is not larger than SL (SB is equal than SL when UB is parallel to LB). When this happens, a point is created inside the coverage area that represents the last trend line value. This point is used as the first value of a new parallelogram area. The SDT only sends the first and the last value for each trend line [9]. Algorithm 1 represent the eight steps of SDT [10].

1) Receive the first point.

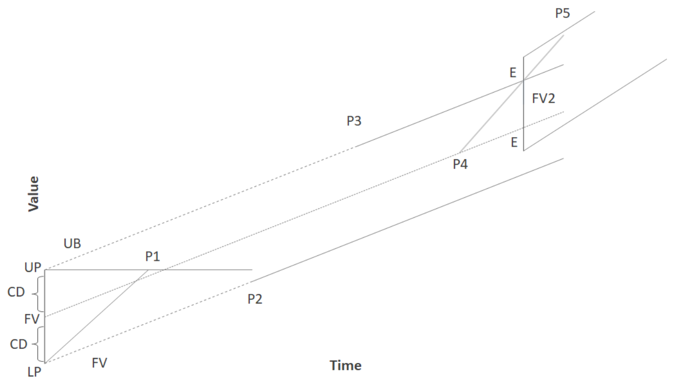


Fig. 1. Graphic representation of SDT.

2) Establish the upper and lower pivot point.
3) Receive the next point.
4) Calculate the current slopes, with respect to the upper and lower pivot.
5) Compare current slopes with prior extreme slopes (SUmax, SLmin). If SUmax is larger than SLmin the algorithm continues in step 6, else return to step 3.
6) When some point is outside to parallelogram, the slope between the last point and the current point is calculated. The crossed boundary is adjusted for being parallel with the other boundary. An interception point between the crossed boundary and the slope is calculated, and a new first point is decided.
7) Broadcast (deliver) the point c as an output point in the compressed data stream of the output signal.
8) Point c is used as the first point of the next segment; the extreme slopes are recalculated; the pivot points are created, and the current point is used again in step 3.

C. Performance metrics

Two metrics are important for assessing the performance of the compression algorithm: compression error (CE) and compression rate (CR). CE measures the relative amount of error observed after compression (Equation 1). It is calculated as the summation of the differences between the uncompressed data (T_i) and the compressed data results after the decompressed process ($T_i^{'}$), divided by the summation of the absolute values of the uncompressed data.

$$CE = \frac{\sum_{i=1}^{n} |T_i - T_i^{'}|}{\sum_{i=1}^{n} |T_i|} \tag{1}$$

CR aims to assess the efficiency of the compression process (Equation 2) and represents the reduction of samples achieved using a compression algorithm. It is calculated as the complement of the division of the compressed samples by the uncompressed ones.

$$CR = 1 - \frac{CompressedSamples}{UncompressedSamples} \tag{2}$$

Algorithm 1 SDT

```
 1: Initialization
 2: $CD = selected\ error$
 3: $c = d = \langle Time, Value \rangle$                          ▷ Step 1
 4: $u = c + \langle 0, CD \rangle$,    $l = c + \langle 0, -CD \rangle$      ▷ Step 2
 5: $s^u_{Max} = -\infty$,     $s^l_{Min} = \infty$
 6: BROADCAST(c)

 7: function NEW_WINDOW()
 8:     $u = c + \langle 0, CD \rangle$                              ▷ Step 8
 9:     $l = c + \langle 0, -CD \rangle$
10:     $s^u = (d_y - u_y)/(d_x - u_x)$
11:     $s^l = (d_y - l_y)/(d_x - l_x)$
12:     $s^u_{max} = s^u$
13:     $s^l_{min} = s^l$
14: end function

15: function SDT($Time, Value$)
16:     $p = d$
17:     $d = \langle Time, Value \rangle$                            ▷ Step 3
18:     $s^u = (d_y - u_y)/(d_x - u_x)$                              ▷ Step 4
19:     $s^l = (d_y - l_y)/(d_x - l_x)$
20:     if $s^u > s^u_{max}$ then                                    ▷ Step 5,upper pivot
21:         $s^u_{max} = s^u$
22:         if $s^u_{max} > s^l_{min}$ then
23:             $s^o = (d_y - p_y)/(d_x - p_x)$                      ▷ Step 6
24:             $c^u = (u_y - p_y + s^o p_x - s^l_{min} u_x)/(s^o - s^l_{min})$
25:             $c = \langle c^u\ ,\ u_y + s^l_{min}(c^u - u_x) - CD/2 \rangle$
26:             BROADCAST(c)                                         ▷ Step 7
27:             NEW_WINDOW()
28:         end if
29:     end if
30:     if $s^l < s^l_{min}$ then                                    ▷ Step 5, Lower pivot
31:         $s^l_{min} = s^l$
32:         if $s^u_{max} > s^l_{min}$ then
33:             $s^o = (d_y - p_y)/(d_x - p_x)$                      ▷ Step 6
34:             $c^l = (l_y - p_y + s^o)p_x - s^u_{max} l_x)/(s^o - s^u_{max})$
35:             $c = \langle c^l\ ,\ l_y + s^u_{max}(c^l - l_x) + CD/2 \rangle$
36:             BROADCAST(c)                                         ▷ Step 7
37:             NEW_WINDOW()
38:         end if
39:     end if
40: end function
```

Note: d is a sample point value, p is the past point value and c is a calculated point by the SDT algorithm. These points are represented as a 2-tuple of the form $< time, value >$.

It is important to note that these metrics should be used together to represent the performance of an algorithm properly. In this sense, this paper proposes a new metric called compression criteria (CC), as illustrated in Eq. 3.

$$CC(CE, CR) = \frac{2 \cdot CR \cdot (1 - CE)}{CR + (1 - CE)} \tag{3}$$

The harmonic mean of the two primary metrics is used in this new metric because it is more sensitive when one value is much smaller than the other. This metric represents the compression efficiency. The set of valid CC values belongs to the range [0,1], where a good compression has a value close to 1, that corresponds to a high CR and low CE values.

III. SSDT proposal: The Self-Definition SDT

The scenario adopted in the proposal just considers IoT devices running an SDT algorithm connected to a server; it is not considered interactions among devices. SDT has as an important challenge: the appropriate selection of CD value. Conventionally, it is necessary to have knowledge about the signal behavior to select a suitable value. In this paper, a self-configuration step is proposed, allowing the definition of an appropriate CD value called Self-definition SDT (SSDT).

As SSDT was designed to run on IoT devices, it is a lightweight CD decision process that executes in just N+1 sensing rounds. The absolute value of the slope between consecutive points is used as the analyzed characteristic (Eq. 4).

$$s^i(d, p) = |\frac{d_y - p_y}{d_x - p_x}| \tag{4}$$

The proposal in this paper includes four versions with different mathematical equations.

The arithmetic mean (MEAN): It is the sum of the sampled values divided by the number of items (Eq. 5).

$$f_{mean} = \frac{1}{n} \sum_{i=1}^{n} Xi \tag{5}$$

Exponential Moving Average (EMA): It is a first-order infinite impulse response filter that applies weighting factors which decrease exponentially (Eq. 6). The α parameter has a value in the interval [0,1].

$$f_{ema}(d, p) = (1 - \alpha)CD + \alpha \cdot S^i(d, p) \tag{6}$$

Mean without zero (ZMEAN): It corresponds to arithmetic mean, but the values equal to zero are ignored, reducing the number of samples used in the mean (Eq. 7).

$$f_{zmean}(d, p) = \begin{cases} CD + \dfrac{s^i(d, p)}{N^z} & \text{if } s^i(d, p) \neq 0 \\ \dfrac{CD \cdot (N^z + 1)}{N^z} & \text{if } S^i(d, p) = 0 \end{cases} \tag{7}$$

Range (RANGE): It corresponds to the difference between the largest and smallest values (Eq. 8).

$$f_{range} = \frac{MAX + MIN}{2} \tag{8}$$

The SSDT code is illustrated in Algorithm 2.

IV. Related Work

In this section, related work is divided into two parts. The first address data compression proposals set in IoT environments and the second focus on SDT algorithms.

A. Data Compression in IoT

The work in [11] proposes a method that reduces the amount of stored data using an approximation method. The method discards the values outside the sensor range, featuring a lossy compression algorithm. The algorithm has low computational complexity but a high error rate, because it may ignore relevant information from the samples.

In [5], it is proposed a coding scheme for delta compression. The main goal is to perform a data compression for IoT solutions in applications with temporally correlated data. It is based on the symbol reductions for representing frequency values and the reduction of redundant information. The messages are only delivered when a threshold of messages is achieved.

Algorithm 2 SSDT

```
 1: Initialization
 2: $d = \langle Time, Value \rangle$
 3: $CD = Count = 0$
 4: $Training_{min} = TRAINING$
 5: $Trained = False$
 6: $\alpha = 0.15$                                        ▷ EMA
 7: $N^z = TRAINING$                                      ▷ Zmean
 8: $Max = -\infty$                                       ▷ Range
 9: $Min = \infty$                                        ▷ Range
10: BROADCAST(d)

11: function SSDT($Time, Value$)
12:     if $Trained == False$ then
13:         TRAIN($Time, Value$)
14:     else
15:         SDT($Time, Value$)
16:     end if
17: end function

18: function TRAIN($Mode, Time, Value$)
19:     $p = d$
20:     $d = \langle Time, Value \rangle$
21:     BROADCAST(d)
22:     if $Mode == MEAN$ then          $CD = $ MEAN$(d, p)$
23:     else if $Mode == ZMEAN$ then    $CD = $ ZMEAN$(d, p)$
24:     else if $Mode == RANGE$ then    $CD = $ RANGE$(d, p)$
25:     else if $Mode == EMA$ then      $CD = $ EMA$(d, p)$
26:     end if
27:     if ++$Count == Training_{min}$ then
28:         $c = d$
29:         $s^u_{Max} = -\infty$
30:         $s^l_{Min} = \infty$
31:         $Trained = True$
32:     end if
33: end function
```

This reduces the overhead caused by the message headers. The results show that this technique can achieve up to 85% energy saving if applied to data that are temporally correlated. The scheme is not recommended in situations that require real-time data.

Hossain [12] proposes a data compression in two steps. The first is done in a fog node, using a lossy algorithm with a compression rate of 50% and the second in the cloud storage using a lossless algorithm. The main goal is to reduce the energy consumption by the transmission of a large quantity of data from the fog node to the cloud.

Deepu [13] proposes a data compression and transmission scheme for power saving in IoT scenarios. It is a hybrid scheme that uses a lossy and a lossless algorithm. It can be used in situations that a lossy compression algorithm can eliminate relevant information and a lossless algorithm does not achieve an acceptable compression rate.

B. Swinging Door Trending

The proposal of [9] is an Adaptive Swinging Door Trending (ASDT), a variation of the traditional SDT. ASDT has the same basic characteristics than SDT but uses a real-time trend analysis as a mechanism to incorporate variations on main parameters. The analysis is made using an exponential moving average (EMA) that works as a filter. A disadvantage of ASDT is the necessity of a previous definition of the EMA parameters.

[14] proposes the Improved Swinging Door Trending (ISDT), which employs an adaptive CD that changes according to the previous compression result. ISDT modifies the parameters dynamically as new data are acquired. ISDT proposes a CD with an upper and lower limits that restrict the range of values that CD can assume. An ISDT disadvantage is that, as SDT, it requires parameters (e.g. upper and lower limit definition) to be user-defined [14].

The propose of [15] is an in-network time-series data compression algorithm called Distributed Swinging Door Trending (DSDT) to be used in IoT devices. The goal is to use the computing resource of the sensor to compress the raw sensing data. The compressing center can self-adapt according to the conditions of current bandwidth and computing resource usage. DSDT has two algorithms, the first to the sensor device and the second to the compressing center.

In [16] is proposed an improved SDT called KSDT algorithm for being used in a wireless sensing for the acquisition of the mechanical failure signal. The authors designed a way to dynamically adjust the value of CD that is based on the specific parameters used in machine condition monitoring. As the algorithm was created for a specific use case, its applicability to other usage scenarios cannot be assumed.

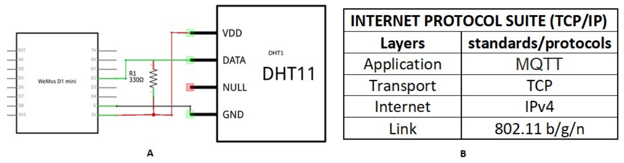
The SDT is proposed as a data compression algorithm in a WSN in [17]. The focus of this research was to investigate the performance of data compression algorithms in monitoring applications for industrial automation environments. The case study was to analyze the performance of a WSN that has sensor nodes equipped with SDT. The results show that with CR around 85% the network life is tripled.

An asynchronous Event-Triggered for Smart Sensor Network Architectures is proposed in [18] to be used on OMG´S standards. It integrated a data compression algorithm into smarts sensors. The results show that a local compression decreases considerably the data exchanged in the communication network.

V. SSDT Assessment

A. Experimental setup

An experimental setup was implemented by using WeMos D1 Wi-Fi UNO ESP-12E boards equipped with temperature and humidity DHT11 sensors as IoT devices [19]. Figure 2 (A) shows the hardware setup and Figure 2 (B) shows the communication protocols that were used in each network layer.



INTERNET PROTOCOL SUITE (TCP/IP)	
Layers	**standards/protocols**
Application	MQTT
Transport	TCP
Internet	IPv4
Link	802.11 b/g/n

Fig. 2. (A) circuit design. (B) TCP/IP model.

The system uses mosquito [20] as MQTT broker and a Python script as a subscribe client to receive the sensor information and manages a mongoDB database.

A database was formed in the LAPESD laboratory at the Federal University of Santa Catarina, Brazil. Two IoT devices (Device 1 and Device 2) sent data with and without SDT algorithm for performance comparison. Data collection was made every 5 seconds, and 192610 temperature and humidity data samples were taken in each device.

As IoT devices have hardware limitations [21], it is important to analyze if they have enough computational resources to compress the data on the fly. At first, analyzing Algorithm 1, SDT computational complexity represented in BigO notation shown to be just O(1). Next, the experimental setup was executed, and the execution times from the WeMos D1 device with a clock of 160 MHz were analyzed. The worst-case execution time obtained in the implementation was 162 microseconds, which can be considered a satisfactory result for the major applications.

In addition to the two IoT devices, a temperature and humidity dataset from Ljubljana and NovoMes stations [22] were also used. All of these collected data were used in a Python algorithm to evaluate the results and to simulate different CD value scenarios.

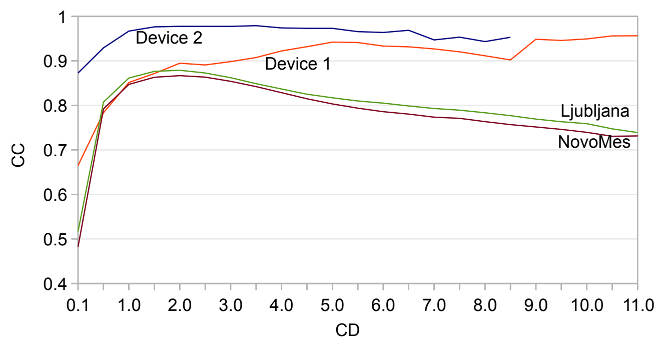


Fig. 3. Impact of CD variation on CC results of temperature sensors.

B. CD selection on each sensor

In SDT methods, the value of the CD should be carefully selected as it depends on the number of samples and the behavior of the sensing device. A CD value that results in high CR and CE values is not a good choice. This situation may represent a CD with high compression but with a high error in signal decompression. The CC metric was introduced in this paper in order to balance the CR and CE metrics. When CD value changes, CR and CE values are affected, as well as CC metric that is formed by combining both.

Figure 3 shows the behavior of the CC in relation to the CD variation of four temperature sensing devices. Devices 1 and 2 are from the LAPESD laboratory, and "Ljubljana" and "NovoMes" are from Slovenia database. Table I shows the best CC values for each device. The goal is to enable self-selection of CD values with adequate values of CC. (Note that the CD values from Figure 3 were obtained by brute force, using all collected samples and varying the CD values at intervals of 0.1, which is not feasible to be done on IoT devices.)

TABLE I
BEST CASES OF CC VALUES.

Source	**Sensor**	**CD**	**CE**	**CR**	**CC**
Temperature	Device 1	10.9	0.080	0.999	0.958
	Device 2	3.5	0.039	0.998	0.978
	Ljubljana	1.9	0.134	0.893	0.879
	NovoMes	2.1	0.151	0.885	0.867
Humidity	Device 1	6.0	0.036	0.998	0.981
	Device 2	5.1	0.032	0.997	0.982
	Ljubljana	54.5	0.151	0.999	0.918
	NovoMes	10.0	0.089	0.891	0.901

C. SSDT results

Tables II to V show the obtained CC for each device using different amount of samples. The objective of assessing the four techniques with different sample quantities is because the number of samples used in the algorithm training influences the performance of the algorithm. A technique that yields a good result with fewer samples (e.g., 100) will be suitable for use in online approaches. Moreover, note that using larger samples does not always guarantee better performance. The best results for each device are highlighted in blue.

In these results, the α value used in EMA algorithm was assumed as 0.125.

TABLE II
VALUES OF CC PER SAMPLES, DEVICE 1.

	Temperature						
	100	**250**	**500**	**1000**	**2500**	**5000**	**10000**
MEAN	0.69	0.67	0.67	0.67	0.67	0.67	0.74
RANGE	0.89	0.89	0.91	0.91	0.92	0.92	0.94
ZMEAN	0.89	0.87	0.89	0.89	0.89	0.89	0.89
EMA	0.67	0.66	0.66	0.67	0.67	0.86	0.86
	Humidity						
	100	**250**	**500**	**1000**	**2500**	**5000**	**10000**
MEAN	0.73	0.78	0.79	0.78	0.79	0.77	0.77
RANGE	0.98	0.97	0.97	0.97	0.97	0.97	0.97
ZMEAN	0.91	0.94	0.95	0.96	0.95	0.93	0.91
EMA	0.71	0.67	0.93	0.59	0.59	0.73	0.78

TABLE III
VALUES OF CC PER SAMPLES, DEVICE 2.

	Temperature						
	100	**250**	**500**	**1000**	**2500**	**5000**	**10000**
MEAN	0.90	0.88	0.88	0.88	0.88	0.88	0.88
RANGE	0.98	0.98	0.98	0.98	0.97	0.97	0.97
ZMEAN	0.98	0.98	0.98	0.98	0.98	0.98	0.98
EMA	0.87	0.87	0.87	0.87	0.90	0.87	0.87
	Humidity						
	100	**250**	**500**	**1000**	**2500**	**5000**	**10000**
MEAN	0.89	0.93	0.91	0.91	0.91	0.91	0.86
RANGE	0.98	0.98	0.98	0.98	0.98	0.98	0.98
ZMEAN	0.97	0.98	0.98	0.98	0.98	0.98	0.97
EMA	0.91	0.86	0.82	0.86	0.89	0.93	0.93

Analyzing the results in Tables II to V, it is possible to assess the behavior of each mathematical formula. Note that, MEAN and EMA had the bad results concerning the CC metric. However, EMA could have better results by tuning, for each scenario, its α value carefully.

TABLE IV
Values of CC per samples, Ljubljana.

	Temperature						
	100	**250**	**500**	**1000**	**2500**	**5000**	**10000**
MEAN	0.83	0.85	0.84	0.83	0.81	0.84	0.85
RANGE	0.88	0.83	0.82	0.81	0.81	0.81	0.81
ZMEAN	0.84	0.85	0.85	0.84	0.83	0.85	0.86
EMA	0.87	0.83	0.86	0.74	0.86	0.83	0.52
	Humidity						
	100	**250**	**500**	**1000**	**2500**	**5000**	**10000**
MEAN	0.78	0.82	0.82	0.79	0.79	0.84	0.85
RANGE	0.88	0.91	0.91	0.91	0.90	0.89	0.88
ZMEAN	0.82	0.86	0.86	0.85	0.85	0.87	0.87
EMA	0.75	0.86	0.88	0.30	0.88	0.87	0.42

TABLE V
Values of CC per samples, NovoMes.

	Temperature						
	100	**250**	**500**	**1000**	**2500**	**5000**	**10000**
MEAN	0.83	0.79	0.83	0.83	0.84	0.85	0.85
RANGE	0.84	0.84	0.78	0.78	0.78	0.78	0.28
ZMEAN	0.85	0.83	0.84	0.83	0.85	0.85	0.85
EMA	0.64	0.85	0.48	0.83	0.83	0.85	0.79
	Humidity						
	100	**250**	**500**	**1000**	**2500**	**5000**	**10000**
MEAN	0.84	0.75	0.80	0.81	0.84	0.86	0.85
RANGE	0.90	0.90	0.90	0.90	0.89	0.88	0.88
ZMEAN	0.87	0.82	0.84	0.84	0.86	0.87	0.87
EMA	0.77	0.60	0.72	0.89	0.89	0.88	0.84

For most scenarios, RANGE and ZMEAN have the best results, obtaining a CE close to 0, a CR close to 1 and, consequently, a CC value close to 1, too. For most scenarios, RANGE achieved the best results.

Table VI includes the CE and CR value when is used the RANGE method. Note that the use of this compression method achieves low error values (CE) and high compression ratio (CR) on the IoT devices.

TABLE VI
Results of the RANGE method in temperature sensors.

	Device 1			**Device 2**			**Ljubljana**			**NovoMes**		
Samples	**CD**	**CR**	**CE**	**CD**	**CR**	**CE**	**CD**	**CR**	**CE**	**CD**	**CR**	**CE**
100	2.0	0.850	0.06	3.0	0.997	0.04	1.8	0.887	0.13	3.7	0.929	0.24
250	2.0	0.850	0.06	3.0	0.997	0.04	4.2	0.946	0.26	3.7	0.929	0.24
500	3.5	0.908	0.09	3.0	0.997	0.04	4.7	0.953	0.28	6.7	0.972	0.35
1000	3.5	0.908	0.09	3.0	0.997	0.04	5.2	0.960	0.29	6.7	0.972	0.35
2500	4.0	0.960	0.12	5.0	0.999	0.05	5.2	0.960	0.29	6.7	0.972	0.35
5000	4.0	0.960	0.12	5.0	0.999	0.05	5.2	0.960	0.29	6.7	0.972	0.35
10000	5.0	0.977	0.09	5.0	0.999	0.05	5.2	0.960	0.29	23.5	0.999	0.84

VI. Final Remarks

In this paper, we have showed a modification of Swinging Door compression algorithm to enhance the performance of IoT devices. The results demonstrated that it is possible to use different mechanisms to the self-definition of the CD parameter. Four mechanisms were used in this proposal, and both RANGE and ZMEAN presented the best results.

Acknowledgements: This work was supported by FAPEPI/MCT/CNPq/CT-INFRA 007/2018 and CAPES Print Program 698503P.

References

[1] J. Uthayakumar, T. Vengattaraman, and P. Dhavachelvan, "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*, 2018.

[2] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, and X. Yang, "A survey on the edge computing for the internet of things," *IEEE Access*, vol. 6, pp. 6900–6919, 2018.

[3] A. R. Biswas and R. Giaffreda, "Iot and cloud convergence: Opportunities and challenges," in *2014 IEEE World Forum on Internet of Things (WF-IoT)*, March 2014, pp. 375–376.

[4] J. Azar, A. Makhoul, M. Barhamgi, and R. Couturier, "An energy efficient iot data compression approach for edge machine learning," *Future Generation Computer Systems*, vol. 96, pp. 168 – 175, 2019.

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

[6] O. MAHDI, M. Mohammed, and A. Jasim Mohamed, "Implementing a novel approach an convert audio compression to text coding via hybrid technique," *IJCSI*, 11 2013.

[7] E. H. Bristol, "Swinging door trending: adaptive trend recording," in *Proc. of the ISA National Conf.*, 1990, pp. 749–753.

[8] K. P. Shravana and D. S. V. Veena, "Review on lossless data compression using x-matchpro algorithm," in *2017 2nd IEEE Int. Conf. on Recent Trends in Electronics, Inf. Comm. Technology (RTEICT)*, May 2017, pp. 1095–1100.

[9] D. C. S. L. A. G. Edson J. M. Neto, Luiz Augusto, "Adaptive swinging door trending: Um algoritmo adaptativo para compressao de dados em tempo real," *Anais do XX Congresso Brasileiro de Automática*, 2014.

[10] E. H. Bristol, "Data compression for display and storage," May 1987, uS Patent 4,669,097.

[11] V. Alieksieiev, "One approach of approximation for incoming data stream in iot based monitoring system," in *2018 IEEE Second Int. Conf. on Data Stream Mining Processing (DSMP)*, Aug 2018, pp. 94–97.

[12] K. Hossain and S. Roy, "A data compression and storage optimization framework for iot sensor data in cloud storage," in *2018 21st Int. Conf. of Computer and Inf. Technology (ICCIT)*, Dec 2018, pp. 1–6.

[13] C. Deepu, C.-H. Heng, and Y. Lian, "A hybrid data compression scheme for power reduction in wireless sensors for iot," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 11, no. 2, pp. 245–254, 2017, cited By 24.

[14] F. Xiaodong, C. Changling, L. Changling, and S. Huihe, "An improved process data compression algorithm," in *Proc. of the 4th World Congress on Intelligent Control and Automation*, vol. 3, June 2002, pp. 2190–2193.

[15] J. K. Zhao, L. S. Mu, H. Ouyang, P. F. Zhu, Y. K. Li, L. H. Yang, and L. J. Chen, "In-network time-series data compression for electric internet of things," in *Applied Mechanics and Materials*, vol. 241, 2013, pp. 3213–3223.

[16] X. M. Liu, S. Han, Y. J. Zhou, and Y. Yao, "An energy-efficiency wireless sensing method for mechanical failure signal," in *Key Engineering Materials*, vol. 572, 2014, pp. 451–454.

[17] I. M. D. Silva, L. A. Guedes, and F. Vasques, "Performance evaluation of a compression algorithm for wireless sensor networks in monitoring applications," in *2008 IEEE Int. Conf. on Emerging Technologies and Factory Automation*, Sep. 2008, pp. 672–678.

[18] E. M. Leao, L. A. Guedes, and F. Vasques, "An event-triggered smart sensor network architecture," in *2007 5th IEEE Int. Conf. on Ind. Inf.*, vol. 1, June 2007, pp. 523–528.

[19] Ai-Thinker, "Esp-12e wifi module," 2015, v1.0. [Online]. Available: https://www.kloppenborg.net/images/blog/esp8266/esp8266-esp12e-specs.pdf

[20] R. A. Light, "Mosquitto: server and client implementation of the mqtt protocol," *Journal of Open Source Software*, 2017.

[21] L. Schrickte, C. Montez, R. De Oliveira, and A. Pinto, "Design and implementation of a 6LoWPAN gateway for wireless sensor networks integration with the internet of things," *International Journal of Embedded Systems*, vol. 8, no. 5-6, pp. 380–390, 2016.

[22] L. Ashcroft, J. R. Coll, A. Gilabert, P. Domonkos, E. Aguilar, J. Sigro, M. Castella, P. Unden, I. Harris, P. Jones, and M. Brunet, "Air temperature and Relative humidity measurements in Slovenia," 2018. [Online]. Available: 10.1594/PANGAEA.887091