Adaptive Lossless Entropy Compressors for Tiny IoT Devices

Massimo Vecchio, *Member, IEEE,* Raffaele Giaffreda, and Francesco Marcelloni, *Member, IEEE*

***Abstract*—Internet of Things (IoT) devices are typically powered by small batteries with a limited capacity. Thus, saving power as much as possible becomes crucial to extend their lifetime and therefore to allow their use in real application domains. Since radio communication is in general the main cause of power consumption, one of the most used approaches to save energy is to limit the transmission/reception of data, for instance, by means of data compression. However, the IoT devices are also characterized by limited computational resources which impose the development of specifically designed algorithms. To this aim, we propose to endow the lossless compression algorithm (LEC), previously proposed by us in the context of wireless sensor networks, with two simple adaptation schemes relying on the novel concept of appropriately rotating the prefix-free tables. We tested the proposed schemes on several datasets collected in several real sensor network deployments by monitoring four different environmental phenomena, namely, air and surface temperatures, solar radiation and relative humidity. We show that the adaptation schemes can achieve significant compression efficiencies in all the datasets. Further, we compare such results with the ones obtained by LEC and, by means of a non-parametric multiple statistical test, we show that the performance improvements introduced by the adaptation schemes are statistically significant.**

***Index Terms*—Internet of Things (IoT), lossless compression, wireless sensor networks, power efficiency.**

I. Introduction

MODERN society yearns for moving towards an *always-connected* paradigm, so as to finally fill in the gap between the physical and the digital worlds. Indeed, networks (both wired and wireless) are (almost) everywhere, open standards (*e.g.*, WiMAX, IPv6 and 6LoWPAN) are defined and rolled out, while several technologies beneath the umbrella of "Future Internet" are already being researched and developed [1]. In this context, the "Internet of Things" (IoT) represents one of the visions that just captured the interest of academia and industry at the time of its coinage (according to Ashton, K., "Internet of Things" started its life since the presentation he made at Procter&Gamble in 1999) and more and more momentum has gained in terms of development efforts in recent years.

The original IoT vision encloses a framework where objects (*i.e.*, *things*) can be uniquely identified and whose virtual representations can be accessed and controlled using an Internet-like structure [2]. Undoubtedly, the recent advances in microelectronics and communications, which are deeply impacting on the costs, sizes and power requirements of the IoT main enabling technologies (*e.g.*, RFIDs, sensors, actuators) have pushed such embryonic vision towards a more established reality. Nowadays, several disparate IoT applications permeate through practically all areas of every-day life of individuals, enterprises, and society as a whole; IoT already extends across industries, including intelligent environmental monitoring, intelligent building, homes, and cities, smart transportation, and smart health care [3], [4]. It is not a coincidence that a prediction from Forrester Research [5] has revealed that global IoT revenues will be thirty times those of the Internet, making it the next trillion-level communication industry linking over a hundred billion devices by 2020 [6].

Despite the plethora of application domains and the specific features each single application might require, a list of common modules characterizing all IoT solutions can be summarized as (see [7]):

- a module for interaction with local IoT devices (*e.g.*, embedded in mobile phones or located in the immediate proximity of the user and reachable through short range wireless interfaces);
- a module for local analysis and processing of observations acquired by IoT devices;
- a module for interaction with remote IoT devices, directly over the Internet or, more likely, through a proxy;
- a module for application-specific data analysis and processing;
- a module for integration of IoT-generated information into the business processes of an enterprise;
- a (web or mobile) user interface for visual representation of measurements in a given context (*e.g.*, maps) and interaction with the user.

Due to the ever increasing amount of different IoT devices used in diverse application domains, wireless networks are largely growing in number and density. Though several IoT devices will be still connected to the energy grid all the time, the great majority of them will have to rely on their own limited energy resources or energy harvesting throughout their lifetime. Thus, slim and lightweight implementations at all layers (*i.e.*, from the communication to the application layers) still represent key-features for effective and more robust IoT applications. Further, considering the exponentially growing scale of IoT deployments, the heterogeneity of the involved devices and the continuously changing scenarios and real-time requirements (operational conditions), flexible design at both network and device levels represents also a desirable and almost unavoidable property.

Manuscript received June 1, 2013; revised September 25, 2013; accepted November 12, 2013. The associate editor coordinating the review of this paper and approving it for publication was E. Koksal.

M. Vecchio and R. Giaffreda are with CREATE-NET, Trento, Italy (e-mail: {massimo.vecchio, raffaele.giaffreda}@create-net.org).

F. Marcelloni is with the Dipartimento di Ingegneria dell'Informazione of the University of Pisa, Italy (e-mail: f.marcelloni@iet.unipi.it).

Digital Object Identifier 10.1109/TWC.2013.121813.130993

In this landscape, wireless sensor networks (WSNs) play one of the most important roles in enabling the IoT paradigm. Further, the benefits of connecting both WSNs and other IoT devices go beyond the simplistic remote access capability, since heterogeneous information systems may be able to collaborate and provide common services [8]. Examples of such services include, for instance, the possibility of pushing data produced by wireless sensor nodes directly to the cloud or social platforms [9], [10]. In particular, Libelium company already commercializes wireless sensor nodes able to send collected sensor data (directly, or through a special node working as an Internet gateway) to Twitter and Wordpress [9], while the Cosm platform (formerly Pachube) is an online database service allowing developers to connect sensor-derived data to the Web and to build their own applications based on that data [10].

By deeply analyzing the breeding ground of existent IoT services and components we realized that, except for the Cosm platform that adopts a general-purpose gzip compression scheme for compressing the sensed data stream before sending it, to the best of our knowledge a lightweight and adaptive compression scheme suitable for the IoT sensing devices is not available in the literature yet. Since the main cause of energy consumption in IoT devices is generally represented by the communication unit, it still makes sense for these devices to reduce the amount of information to be transmitted by means of compression techniques [11]. Moreover, some IoT devices may be featured by even more reduced hardware capabilities (*i.e.*, memory and microprocessor) than currently available wireless sensor nodes. Thus, special attention should be paid when designing software solutions suitable for such devices, in terms of memory occupation and computational effort requirements. In the framework of WSNs, the Lossless Entropy Compression (LEC) algorithm, we introduced in [11], [12], has proven to be very effective in compressing environmental data and also very efficient in terms of power consumption [13].

Nevertheless, in the most general IoT framework, a compression algorithm should be able to react to highly variable operational conditions. Thus, we believe that the LEC algorithm can be improved by adding some mechanism for making it adaptive. To this aim, we endow LEC with two simple adaptation schemes, which allow it to cope with variable operational conditions, though preserving its original simplicity. We tested the proposed schemes on several datasets collected in several real sensor network deployments. We show that the adaptation schemes can achieve significant compression efficiencies in all the datasets. Further, by means of a non-parametric multiple statistical test we present that the two schemes outperform in terms of compression efficiency LEC, while there exists no statistical difference between them.

The paper is organized as follows. Sec. II reviews the LEC algorithm and some of its variants proposed in the recent literature. In Sec. III we introduce the two adaptation schemes. Sec. IV discusses the implementation of the LEC algorithm and of the adaptation schemes. The experimental analysis is given in Sec. V. Finally, we draw some conclusions in Sec. VI.

II. Related Works

In this section, we first describe the LEC algorithm and then we discuss some of the variants which have been proposed in the literature to make it adaptive.

A. The LEC algorithm

The main idea of the LEC algorithm relies on dividing the alphabet of numbers into groups whose sizes increase exponentially. Indeed, like in Golomb and Elias codings [14], [15], a LEC codeword is a hybrid of unary and binary codes, where the unary code (a variable-length code) specifies the group and the binary code (a fixed-length code) represents the index within the group. The main difference between LEC and its precursors is that groups are entropy coded rather than unary coded. For more details on the LEC algorithm and its similarities and peculiarities with respect to Golomb and Elias codings, the interested reader can refer to [11].

In the sensing unit of a sensor node, each measurement m_i acquired by a sensor is converted by an ADC to a binary representation r_i on R bits, where R is the resolution of the ADC *i.e.*, the number 2^R of discrete values the ADC can produce over the range of analogue values. For each new acquisition r_i, LEC computes the difference $d_i = r_i - r_{i-1}$, which is input to an entropy encoder (to compute d_1 it is conventionally assumed that $r_0 = 0$). The entropy encoder performs compression losslessly by encoding differences d_i more compactly based on their statistical characteristics. In particular, each non-zero d_i value is represented as a bit sequence bs_i composed of two parts $s_i|a_i$, where s_i codifies the number n_i of bits needed to represent d_i (*i.e.*, the group d_i belongs to) and a_i is the representation of d_i (*i.e.*, the index position in the group). When d_i is equal to 0, the corresponding group (*i.e.*, group 0) has size equal to 1 and therefore there is no need to codify the index position in the group: it follows that a_i is not represented.

Formally, n_i is computed as

$$n_i = \begin{cases} 0 & \text{if } d_i = 0 \\ \lfloor \log_2 (|d_i|) \rfloor + 1 & \text{otherwise.} \end{cases} \quad (1)$$

It follows that at most n_i is equal to R. Thus, in order to encode n_i a prefix-free table $\mathcal{S}$ of $R + 1$ entries has to be specified. In principle, $\mathcal{S}$ should depend on the distribution of the differences d_i: more frequent differences should be associated with shorter codes s_i. Since LEC was introduced with the main aim of compressing environmental signals, which vary slowly and therefore are characterised by having the most frequent differences d_i close to 0, we adopted the prefix-free table shown in Table I.

The a_i part of the bit sequence bs_i is a variable-length integer code generated as follows:

$$a_i = \begin{cases} \text{not needed} & \text{if } d_i = 0 \\ \langle d_i \rangle_{n_i}^{L} & \text{if } d_i > 0 \\ \langle d_i - 1 \rangle_{n_i}^{L} & \text{otherwise,} \end{cases} \quad (2)$$

where the operator $\langle x \rangle_y^L$ (resp., $\langle x \rangle_y^H$) indicates the y low-order (resp., high-order) bits of the two's complement representation of the argument x. The procedure used to generate a_i

TABLE I
DEFAULT LEC PREFIX-FREE TABLE.

n_i	s_i	d_i
0	00	0
1	010	$-1, +1$
2	011	$-3, -2, +2, +3$
3	100	$-7, \ldots, -4, +4, \ldots, +7$
4	101	$-15, \ldots, -8, +8, \ldots, +15$
5	110	$-31, \ldots, -16, +16, \ldots, +31$
6	1110	$-63, \ldots, -32, +32, \ldots, +63$
7	11110	$-127, \ldots, -64, +64, \ldots, +127$
8	111110	$-255, \ldots, -128, +128, \ldots, +255$
9	1111110	$-511, \ldots, -256, +256, \ldots, +511$
10	11111110	$-1023, \ldots, -512, +512, \ldots, +1023$
11	111111110	$-2047, \ldots, -1024, +1024, \ldots, +2047$
12	1111111110	$-4095, \ldots, -2048, +2048, \ldots, +4095$
13	11111111110	$-8191, \ldots, -4096, +4096, \ldots, +8191$
14	111111111110	$-16383, \ldots, -8192, +8192, \ldots, +16383$

guarantees that all possible values have different codes. Once bs_i is generated, it is appended to the bitstream which forms the compressed version of the sequence of measurements m_i.

Since its first appearance, the LEC algorithm has captured the attention of several researchers who have proposed different modifications of its original scheme to the aim of improving its performances. The most interesting LEC variants will be reviewed in the next section. As discussed in [16], LEC has also been implemented in hardware on a small, low cost and low power Complex Programmable Logic Device (CPLD) from the Xilinx CoolRunner-II family [17]. Then, the authors have connected the programmed CPLD to an MDA100 sensor node of the Crossbow family [18] to quantify the energy saving introduced by delegating the compression of environmental data samples collected by the sensors on board the MDA100 platform to the CPLD. In particular, they have shown that, by off-loading the compression task to the CPLD, the overall energy consumption of a WSN framework could be halved with respect to not compressing the samples, thus confirming both the effectiveness and the simplicity of the LEC algorithm.

B. LEC variants

Some researchers have tried to improve the LEC algorithm by mitigating its main drawback, that is, the lack of adaptiveness. To better review the most interesting LEC variants, in the following discussion we will refer to the block diagram of the most general LEC-based compression scheme shown in Fig. 1. Here, the prediction block $PRED$, exploiting one or more previous samples, generates the predicted value $\widehat{r}_i$ of the current sample r_i with respect to which the general difference $d_i = r_i - \widehat{r}_i$ is computed; the encoder block ENC compresses d_i by exploiting a prefix-free table $\mathcal{S}_i$; the adaptation block $ADAPT$ is responsible for adapting the prefix-free table to the signal to be compressed. In the original LEC scheme, $PRED$ is implemented as a simple delay block (*i.e.*, the predicted sample $\widehat{r}_i$ coincides with the previous sample r_{i-1}), ENC is responsible for producing the bs_i values as described by eq. (1) and eq. (2), and $ADAPT$ is omitted (*i.e.*, the adaptation block is not implemented by LEC). It follows that $\mathcal{S}_1$ is the default prefix-free table shown in Table I and it remains unchanged during the overall compression task (*i.e.*, $\mathcal{S}_i = \mathcal{S}_1 \ \forall i$).

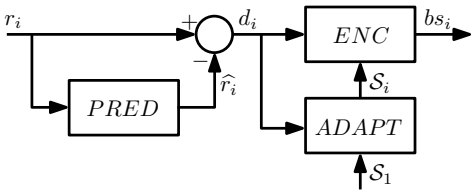


Fig. 1. Block diagram of the most general LEC-based compression scheme.

A simple modification of the LEC algorithm was discussed in [19], where the authors proposed to substitute the delay block used in the prediction block $PRED$ of LEC with a more sophisticated median predictor block. The ENC and $ADAPT$ blocks are unchanged with respect to the LEC scheme. More in detail, this variant of LEC exploits the three previous samples r_{i-3}, r_{i-2} and r_{i-1} for estimating a prediction value $\widehat{r}_i$ that allows reducing the magnitude of d_i. Thus, d_i can be represented more compactly. Then, the encoder adds a 2-bits-long pre-code before each compressed sample to inform the decoder on the actual prediction value used to compress it. We have implemented the variant of LEC and have proved it on the same datasets used in the paper that introduced the original version of LEC [11]. We have obtained 51.02% and 44.95% average compression ratios for the variant against 62.27% and 56.20% for the original LEC on the temperature and relative humidity datasets, respectively. Thus, we can conclude that the compression performances obtained by this variant are actually worse than the ones achieved by the original LEC on real environmental datasets.

Another LEC variant was discussed in [20], where the authors proposed to endow the original LEC algorithm with three different prefix-free tables to better fit the statistics of the different signals to be compressed. Unlike LEC, which compresses each sample on the fly, this variant first collects a sequence of samples; then, it computes all the differences between consecutive samples in the sequence and compresses these differences by using all the three prefix-free tables; finally, it appends the smallest compressed sequence to the compressed bitstream, together with a pre-code that identifies the specific table used to compress the sequence (this pre-code is needed by the decoder to unambiguously recover the compressed block). Referring to Fig. 1, we note that this variant implements the $ADAPT$ block as a 3-way switch which activates the best table among the prefix-free tables after having tested all of them over the sequence of samples to be compressed. In [20], the authors show that this variant outperforms LEC (on average, the compression ratio of the variant is 2-3% higher than the original LEC). On the other hand, this variant requires to store a sequence of samples and three prefix-free tables rather than only one as in the original LEC. Further, it needs to repeat for each sample three times the execution of most of the instructions of the original LEC. Thus, this variant trades a light improvement in the compression performances for the light memory load, the intrinsic on-the-fly elaboration and the low computational complexity of the original LEC.

Another LEC variant was discussed in [21], where the authors have proposed to use the Dynamic Huffman Com-

pression (DHC) scheme [22] to adaptively compress groups n_i. Hence, like in LEC, each Huffman codeword uniquely represents a group n_i of input symbols rather than only one symbol as in the original DHC. Unlike in LEC, which employs the static prefix-free Table I, however, this variant exploits the original Vitter's implementation and management of the tree-structure [23] to codify more frequent groups of symbols with shorter codewords. On the one hand, exploiting the powerful Vitter's implementation to manage the prefix-free table offers a higher degree of adaptiveness, which can be particularly useful when compressing large and high-variable signals. On the other hand, the management of the tree-structure can be computationally intensive and not suitable for the sensor nodes, as observed in [13].

III. The Proposed Adaptive Compression Schemes

Sec. II-A has highlighted how LEC strongly relies on the assumption of knowing the statistics of the signals to be compressed. In particular, its default prefix-free table (Table I) maximally exploits the assumption that the probabilities of the input symbols to the encoder (*i.e.*, the differences of two consecutive samples) are inversely proportional to their absolute values. It follows that, when such assumption does not hold or scarcely holds, it is likely that LEC will not *optimally*[1] compress the input signal. In [11] we discussed the possibility of performing a preliminary study on the specific phenomenon to monitor/compress, so as to off–line identify the optimal prefix-free table. However, we were conscious that this best-effort practice was just a viable workaround for small and homogeneous scenarios, while adaptive mechanisms would have been always preferable.

In this paper we propose two tiny adaptation mechanisms that can be embedded in the LEC approach without affecting its original simplicity. Indeed, in order to effectively enable lossless compression in tiny sensor nodes, the compression algorithm cannot be complex, as the benefits of reducing the use of the radio transceiver on-board the node could be vanished by an intense use of the processing unit [24]. The proposed two mechanisms implement the adaptation by rotating the prefix-free table depending on two different conditions. To make the rotation effective, however, the table has to be accurately defined. In this section, we first introduce how the rotation can be managed. Then, we propose a heuristic approach to adequately generate the prefix-table. Finally, we explain how the rotation is applied in the two mechanisms to adapt the prefix-table to the actual samples to be compressed.

A. Managing rotation of the prefix-free tables

We handle the prefix-free table as a circular buffer. To this aim, we introduce a simple data structure $\bar{\mathcal{S}}$ denoted *prefix-free rotation table*, consisting of a static table $\mathcal{S}$, where the set of T prefix-free codes $s[0], \ldots, s[T-1]$ are actually stored, and an integer vector e of size T, whose elements are initially set to the corresponding position in the vector (*i.e.*, $e[0] = 0, \ldots, e[T-1] = T-1$). Two basic operations can be performed on $\bar{\mathcal{S}}$, namely *access* the k-th entry (*i.e.*, $\bar{s}[k]$) and *rotate* all the elements by h steps. The first operation simply corresponds to the indirection $\mathcal{S}[e[k]]$. The second operation is implemented by applying $(e[k]-h)$ modulo T to all $e[k]$ (*i.e.*, $\forall k \in [0, \ldots, T-1], e[k] = |e[k]-h|_T$, where $|\cdot|_T$ denotes modulo T). Since we access the codes in table $\mathcal{S}$ by the indirection $\mathcal{S}[e[k]]$, the rotation of the elements of e allows us to efficiently simulate the rotation of the prefix-free table $\mathcal{S}$ without changing its elements. Note that only elementary instructions on integer operands are executed in both the operations. Further, we have only introduced an additional integer vector of size T with respect to LEC. In Sec. IV we will describe an even more efficient implementation of the rotation prefix-free table, which actually does not need vector e to manage the access and the rotation.

Although in principle the initialization of the static table $\mathcal{S}$ can be left to the user, in the following we propose a simple heuristic methodology to initialize it in the specific context of adaptive compression of environmental data. Specifically, the goal is to build an effective $\mathcal{S}$ (in terms of compression performances of the algorithms that will rely on it) starting from a set of T prefix-free codes. To avoid unnecessary overloading of notation, we assume that the input prefix-free codes are sorted in ascending order of lengths and relabel them as $s_0, \ldots, s_{T-1}$, such that $len(s_k) \leq len(s_{k+1})\ \forall k$ and $len(x)$ is a function returning the number of bits of x. The proposed heuristic methodology initializes the first entry of $\mathcal{S}$ to s_0 and then adopts the following iterative procedure to initialize the remaining $T-1$ entries: the second entry of $\mathcal{S}$ is set to s_1, the last to s_2, the third to s_3, the second-to-last to s_4, and so on. More formally:

$$s[k] = \begin{cases} s_0 & \text{if } k = 0 \\ s_{2k-1} & \text{if } 0 < k < \lceil T/2 \rceil \\ s_{T-1} & \text{if } k = \lceil T/2 \rceil \\ s_{2(T-k)} & \text{if } \lceil T/2 \rceil < k < T. \end{cases} \tag{3}$$

The proposed heuristic initialization of $\mathcal{S}$ guarantees that initially the shortest input prefix-free code will be assigned to the first entry of the table (recalling that it is a rotation table, we will refer to the shortest prefix-free code as the *center* of the table) while the length of the prefix-free codes assigned to the remaining entries increases with the increase of their distances from the center. The advantages of such property will be clear in the next section. As an example of generation of a table $\mathcal{S}$, let us assume to use the prefix-free codes of the default LEC table (hence, the input prefix-free codes are the s_i entries shown in Table I and $T = 15$). The resulting rotation table is shown in Table II, where the *center* of the table is marked in bold (we recall that at the beginning $\bar{s}[k] = \mathcal{S}[e[k]] = s[k]$).

B. Enabling adaptiveness by prefix-free table rotations based on conditions

The adaptation mechanisms proposed in this paper employ the prefix-free rotation table $\bar{\mathcal{S}}$ to efficiently encode groups n_i of input differences d_i. Adaptiveness is performed by appropriately rotating all the elements by h steps upon verification of specific conditions or, more formally:

[1] in the entropy meaning *i.e.*, compressing the most probable symbols using the shortest codewords.

TABLE II
ROTATION TABLE USING THE PREFIX-FREE CODES OF THE DEFAULT LEC TABLE AS INPUT FOR THE INITIALIZATION: THE CENTER OF THE TABLE IS MARKED IN BOLD.

n_i	$\bar{s}_i$	d_i
0	**00**	0
1	010	$-1, +1$
2	100	$-3, -2, +2, +3$
3	110	$-7, \ldots, -4, +4, \ldots, +7$
4	11110	$-15, \ldots, -8, +8, \ldots, +15$
5	1111110	$-31, \ldots, -16, +16, \ldots, +31$
6	111111110	$-63, \ldots, -32, +32, \ldots, +63$
7	11111111110	$-127, \ldots, -64, +64, \ldots, +127$
8	111111111110	$-255, \ldots, -128, +128, \ldots, +255$
9	1111111110	$-511, \ldots, -256, +256, \ldots, +511$
10	11111110	$-1023, \ldots, -512, +512, \ldots, +1023$
11	111110	$-2047, \ldots, -1024, +1024, \ldots, +2047$
12	1110	$-4095, \ldots, -2048, +2048, \ldots, +4095$
13	101	$-8191, \ldots, -4096, +4096, \ldots, +8191$
14	011	$-16383, \ldots, -8192, +8192, \ldots, +16383$

TABLE III
PREFIX-FREE ROTATION TABLE II, AFTER APPLYING A ROTATION OF $h = 5$ STEPS.

n_i	$\bar{s}_i$	d_i
0	11111110	0
1	111110	$-1, +1$
2	1110	$-3, -2, +2, +3$
3	101	$-7, \ldots, -4, +4, \ldots, +7$
4	011	$-15, \ldots, -8, +8, \ldots, +15$
5	**00**	$-31, \ldots, -16, +16, \ldots, +31$
6	010	$-63, \ldots, -32, +32, \ldots, +63$
7	100	$-127, \ldots, -64, +64, \ldots, +127$
8	110	$-255, \ldots, -128, +128, \ldots, +255$
9	11110	$-511, \ldots, -256, +256, \ldots, +511$
10	1111110	$-1023, \ldots, -512, +512, \ldots, +1023$
11	111111110	$-2047, \ldots, -1024, +1024, \ldots, +2047$
12	11111111110	$-4095, \ldots, -2048, +2048, \ldots, +4095$
13	111111111110	$-8191, \ldots, -4096, +4096, \ldots, +8191$
14	1111111110	$-16383, \ldots, -8192, +8192, \ldots, +16383$

$$\bar{\mathcal{S}}_{i+1} = \begin{cases} \bar{\mathcal{S}}_i.rotate(h) & \text{if } C = TRUE \\ \bar{\mathcal{S}}_i & \text{otherwise.} \end{cases} \quad (4)$$

When a sample r_i is input to the compressor, the LEC's encoding block is used to produce the corresponding bit sequence bs_i, after computing the input difference $d_i = r_i - r_{i-1}$. Unlike LEC, which adopts the prefix-free table shown in Table I for encoding each sample r_i, the adaptive schemes use the prefix-free rotation table $\bar{\mathcal{S}}_i$, which is updated at each sample. In particular, $\bar{\mathcal{S}}_i$ is used to codify the group n_i to which d_i belongs to and thus generate the s_i part of bs_i; then eq. (2) is applied to generate a_i. Once bs_i has been computed, the adaptation block evaluates condition C: if the condition is true, *(i)* a rotation of $h = (n_i - c)$ modulo T steps is performed to produce $\bar{\mathcal{S}}_{i+1}$ and *(ii)* c is updated as $c = n_i$; otherwise, $\bar{\mathcal{S}}_{i+1} = \bar{\mathcal{S}}_i$. We observe that the effect of the rotation is to encode group n_i with the shortest prefix-code (*i.e.*, $\bar{s}[c]$) from the next sample, while all the other elements are rotated consequently, so as to maintain the rigidity of the data structure.

Although different conditions C can be defined to design a plethora of adaptation blocks, in this paper we adopt two simple conditions, denoted $C1$ and $C2$ in the following, for respectively implementing a greedy adaptation and an adaptation based on use frequency. Hence, we introduce two different adaptation blocks, namely GA-LEC and FA-LEC, which differ from each other only for the specific condition to verify (C1 for GA-LEC and C2 for FA-LEC). Both the blocks employ an integer c, which acts as an index of the prefix-free rotation table constantly pointing to its center (hence, at the beginning $c = 0$). $C1$ is implemented as a naive short-circuit, hence $C1$ is always verified in GA-LEC. Though conceptually simple, this choice has a rational justification. Indeed, if r_i represents the first sample of a sequence characterized by differences between consecutive samples in the range of group n_i (or in ranges of groups close to group n_i), GA-LEC will be able to optimize the output bit sequence in terms of number of bits from the next sample r_{i+1}. On the other hand, if r_i is due only to a sample quite different from the average, the adaptation mechanism will bring back the prefix-free rotation table to the previous configuration just at the subsequent sample r_{i+1}. In the latter case, only sample r_{i+1} is possibly coded with a larger number of bits than the one generated by using the previous table used for encoding r_i.

Condition $C2$ is more sophisticated than $C1$ and, consequently, also more complex to support. Indeed, it requires an integer vector f of length T, whose components are initialized to 0. After processing each r_i and producing the corresponding bs_i, the element of f in position n_i (as usual, n_i represents the group which r_i belongs to), is increased by 1, i.e., $f[n_i] = f[n_i] + 1$. Condition $C2$ tests if $f[n_i] \geq f[c]$. Thus, the basic idea of FA-LEC is to rotate $\bar{\mathcal{S}}_i$ only when the current most frequent group changes from c to n_i.

In the following, we give a practical example of application of both the adaptation blocks. Without loss of generality, let us assume that $\bar{\mathcal{S}}_i$ corresponds to the one shown in Table II. Let us suppose that the input symbol to the encoder is $d_i = 31$, which belongs to group $n_i = 5$, and $c = 0$. Then, according to Table II, group $n_i = 5$ is entropy coded as 1111110, while the index position of the symbol in the group is mapped to 11111. Thus, $bs_i = 1111110|11111$ is generated as output. So far, except for the different prefix-free codes used to represent n_i, LEC and the proposed adaptive variants perform the same actions. Indeed, the adaptation mechanisms act at this moment.

As regards GA-LEC, the condition is always verified; hence c is set to 5 and the prefix-free table is rotated of $h = 5$ steps. As regards FA-LEC, $f[5]$ is increased by 1 and the same actions as for GA-LEC are performed only if condition $f[5] \geq f[0]$ is verified. The rotated table of $h = 5$ steps is shown in Table III and, according to it, if the subsequent difference d_{i+1} still belongs to group 5, then the s_{i+1} part of bs_{i+1} will be coded as 00 rather than 110 of the standard LEC. Also, if this difference belongs to groups greater than 5, but close enough to it (*i.e.*, groups 6, 7, 8), the corresponding s_{i+1} part would be 3 bits long, thus saving 1, 2 and 3 bits respectively, with respect to LEC. On the other hand, if d_{i+1} is 0, it would be coded using 8 bits, rather than 2 bits of the standard LEC (recall that, in this case, a_{i+1} is not needed).

Independently of the conditions used in the adaptation blocks, the rotations are consistently applied to the overall prefix-free table, thus affecting the representation of all groups.

In particular, also groups that are close to (resp., far from) the center of the rotation table will be represented by short (resp., large) prefix-codes. Thus, similar to LEC, GA-LEC and FA-LEC are still able to efficiently compress those signals whose distributions of the differences of consecutive samples resemble bell-shaped curves with rather pronounced peak values. The advantage with respect to the standard LEC is that they well perform also when the peak of the bell is displaced with respect to zero. On the other hand, the flatter the bell of the input distribution will be, the less the payoff of the proposed adaptation blocks will be. Indeed, as highlighted in the example, if the next difference d_{i+1} is 0, it would be compressed using 8 bits.

To limit the effects of not efficiently encoding those input differences belonging to groups far from the center of the rotation table, we can adopt two rotation sub-tables in place of the rotation table of size T. The two sub-tables, named $\bar{\mathcal{S}}^L$ and $\bar{\mathcal{S}}^H$, are responsible for encoding groups $0, \ldots, \lceil T/2 \rceil - 1$, and $\lceil T/2 \rceil, \ldots, T-1$, respectively. Let $s_0, \ldots, s_{T-1}$ be the input prefix-free codes sorted in ascending order of lengths. Sub-tables $\bar{\mathcal{S}}^L$ and $\bar{\mathcal{S}}^H$ are initialized by applying eq. (3) to, respectively, the $\lceil T/2 \rceil$ shortest prefix-codes (*i.e.*, $s_0, \ldots, s_{\lceil T/2 \rceil - 1}$) and the remaining prefix-codes (*i.e.*, $s_{\lceil T/2 \rceil}, \ldots, s_{T-1}$). When a sample r_i is input to the compressor, the group n_i which the difference $d_i = r_i - r_{i-1}$ belongs to, determines the sub-table to be used (if $n_i \in [0, \ldots, \lceil T/2 \rceil - 1]$, then $\bar{\mathcal{S}}^L$ will be used, otherwise $\bar{\mathcal{S}}^H$). The sub-tables are handled independently of each other: obviously, only the table with the group n_i will be rotated, if the condition C is satisfied. in the following, we denote the versions of GA-LEC and FA-LEC, which adopt the two rotation tables in place of only one, as GAS-LEC and FAS-LEC, respectively.

Referring to the numerical example discussed above, let us assume that the input symbol to the encoder is $d_i = 31$, which belongs to group $n_i = 5$, and $c^L = 0$ and $c^H = 0$, where c^L and c^H are the two integer indexes pointing to the centers of $\bar{\mathcal{S}}^L$ and $\bar{\mathcal{S}}^H$, respectively. Then, only sub-table $\bar{\mathcal{S}}^L$ is involved in the compression process. If the condition we are considering is verified, the sub-table is rotated by $h = 5$ steps. Table IV shows the two sub-tables after the rotation. If the subsequent difference d_{i+1} still belongs to group $n_i = 5$, then the s_{i+1} part of bs_{i+1} will be coded as 00 rather than 110 of the standard LEC. Moreover, by using the two rotation sub-tables, if d_{i+1} is 0, it would be coded using 3 bits rather than the 8 bits utilized when adopting a unique rotation table.

To highlight the advantage of using two rotation sub-tables for compressing environmental data, Fig. 2 shows, as an example, the distribution of the input differences and the corresponding group distribution of a real temperature dataset[2]. As expected the distribution of the input differences resembles a bell-shaped curve centered around $d_i = 0$ (around the 3% of the input differences are 0 and they will be encoded by LEC using the shortest prefix-free code). On the other hand, the most frequent group is $n_i = 5$. Further, significant components of groups $n_i = 0, \ldots, 8$ make the distribution of the groups rather flat. Thus, there exists a non-negligible probability that, for instance, after a sample in group 5, the subsequent sample is in group 0. As we have already pointed out, in the case of a unique rotation table, when it is centered on group 5, subsequent input differences equal to 0 will be encoded using 8 bits; in the case of two rotation sub-tables, only 3 bits are necessary. Finally, Fig. 3 shows a comparison of the distributions of the prefix-code lengths using LEC, FA-LEC and FAS-LEC. It can be observed that both the versions were able to increase the percentage of input differences encoded by using the shortest prefix-free code up to 22%. Since the adaptive schemes use shorter prefix-free codes for a larger percentage of samples, they both improve LEC. The difference between the two adaptation versions is revealed by observing the right tails of the corresponding distributions of the prefix-free lengths. Indeed, around the 3% of the input differences were encoded using 8 bits by the version with the unique rotation table, while the version with two rotation sub-tables has never needed to resort to large prefix-free codes. In Sec. V, we will observe how this apparently not significant difference will slightly affect the compression performances of the corresponding algorithms. Finally, in the next section we will see that the implementation of the adaptive block based on two sub-tables is minimally different with respect to the one based on a unique table, and that both of them need only some basic extra logic with respect to the original LEC algorithm.

TABLE IV
THE TWO SUB-TABLES AFTER A ROTATION OF $h = 5$ STEPS. NOTICE THAT ONLY THE $\bar{\mathcal{S}}^L$ SUB-TABLE HAS BEEN AFFECTED BY THE ROTATION.

n_i	$\bar{s}_i$	d_i
0	110	0
1	11110	$-1, +1$
2	1110	$-3, -2, +2, +3$
3	101	$-7, \ldots, -4, +4, \ldots, +7$
4	011	$-15, \ldots, -8, +8, \ldots, +15$
5	**00**	$-31, \ldots, -16, +16, \ldots, +31$
6	010	$-63, \ldots, -32, +32, \ldots, +63$
7	100	$-127, \ldots, -64, +64, \ldots, +127$
8	**111110**	$-255, \ldots, -128, +128, \ldots, +255$
9	1111110	$-511, \ldots, -256, +256, \ldots, +511$
10	111111110	$-1023, \ldots, -512, +512, \ldots, +1023$
11	11111111110	$-2047, \ldots, -1024, +1024, \ldots, +2047$
12	111111111110	$-4095, \ldots, -2048, +2048, \ldots, +4095$
13	1111111110	$-8191, \ldots, -4096, +4096, \ldots, +8191$
14	11111110	$-16383, \ldots, -8192, +8192, \ldots, +16383$

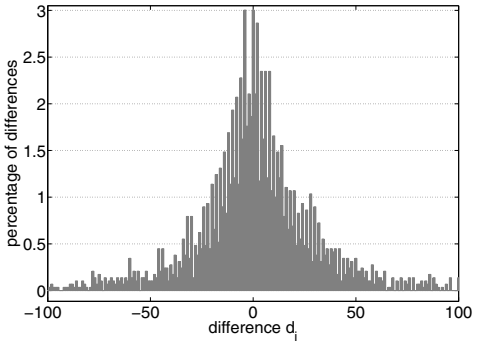
IV. IMPLEMENTATION

In this section we will analyze in detail the implementation of the adaptive schemes. We will start by showing the detailed implementation of the LEC algorithm, by means of pseudo-codes. Then, we will highlight the portions of the LEC code which have to be changed in order to implement the four proposed adaptation schemes. We aim to make evident how the difference between the LEC algorithm and its adaptive versions is limited to very few instructions.

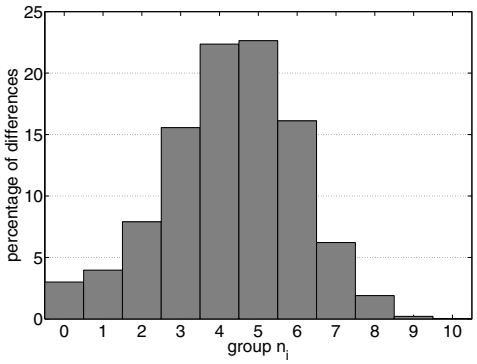
First of all, it is worth to introduce the dictionary structure (namely, $dict$ in the next figures), which is the basic structure LEC relies on to produce the group code (namely, the s_i part of bs_i described in Sec. II-A). More in detail, $dict$ is a structure composed by two constant[3] array (namely, S and

[2]source: air temperature dataset collected by node 9 of the Sensorscope's PDG deployment [25].

[3]For the sake of readability, in the following, we will use capital letters to denote constants.



(a) Distribution of the input differences d_i.



(b) Distribution of groups n_i.

Fig. 2. Example of distributions of (a) the input differences d_i and (b) the corresponding groups n_i of a real temperature dataset.

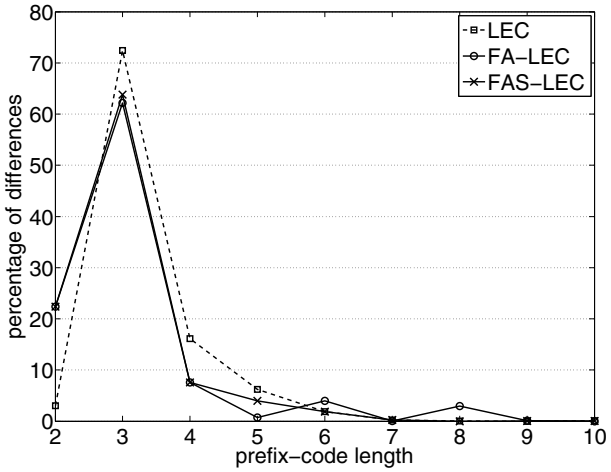


Fig. 3. Comparison of the distributions of the prefix-code lengths using LEC, FA-LEC and FAS-LEC.

L) containing the binary representation of the groups and their actual lengths, respectively. It follows that, for instance, the unary code of group k is represented by the $dict.L[k]$ high-order bits of $dict.S[k]$, that is $\langle dict.S[k] \rangle^H_{dict.L[k]}$.

Thus, starting from Fig. 4, for each input difference di, LEC computes the group ni which di belongs to, by applying eq. (1) in its iterative version (see GROUPOF of Fig. 5). Then, the codeword ci is built and returned by the EMITCODE function (see Fig. 6, where the operators $>>$, $<<$ and $|$ represent the bitwise right shift, left shift and OR, respectively). More in detail, ci is a structure consisting of a 32-bits and an 8-bits unsigned integers (namely, r and l), where r contains the actual representation of the codeword (*e.g.*, the bs_i described in Sec. II-A) in its l high-order bits. Hence, the actual codeword $bs_i = \langle ci.r \rangle^H_{ci.l}$ is appended to the current bitstream ($stream$ in Fig. 4) by the APPEND primitive procedure.

1: **procedure** LEC(di)
2: $ni \leftarrow$GROUPOF(di)
3: $ci \leftarrow$EMITCODE(di, ni)
4: APPEND($stream, \langle ci.r \rangle^H_{ci.l}$)
5: **end procedure**

Fig. 4. The pseudo-code of the LEC compression algorithm.

6: **function** GROUPOF(di)
7: $ni \leftarrow 0$
8: **if** $di < 0$ **then**
9: $di \leftarrow -di$
10: **end if**
11: **while** $di > 0$ **do**
12: $di \leftarrow di >> 1$
13: $ni \leftarrow ni + 1$
14: **end while**
15: **return** ni
16: **end function**

Fig. 5. The pseudo-code of the iterative version of the group computation function (see eq. (1)). This function is executed by LEC and by the four proposed adaptation schemes.

17: **function** EMITCODE(di, ni)
18: $ind \leftarrow ni$
19: $ci.l \leftarrow ni + dict.L[ind]$
20: **if** $ni == 0$ **then**
21: $ci.r \leftarrow dict.S[ind]$
22: **else**
23: **if** $di < 0$ **then**
24: $di \leftarrow di - 1$
25: **end if**
26: $si \leftarrow dict.S[ind]$
27: $ai \leftarrow (di << (32 - ni)) >> dict.L[ind]$
28: $ci.r \leftarrow si|ai$
29: **end if**
30: **return** ci
31: **end function**

Fig. 6. The pseudo-code of the function used by LEC to produce the compressed version of the input difference di as a codeword structure ci.

As introduced in Sec. III, to efficiently manage the access and the rotation of a prefix-free rotation table $\bar{S}$, we do not need the integer vector e. Instead, the access to a specific entry k can be performed by wisely displacing it with respect to the current center of the table (modulo the table size) while a table rotation of h steps is simply performed by updating its current center with h. Thus, from an implementation point of view, we only need to know the table size (*e.g.*, by adding the constant field $dict.T = T$), to keep track of the current center of $\bar{S}$ (*e.g.*, by using the integer field $dict.center$) and to add some basic logic to the original EMITCODE function. More specifically,

the added logic is shown in Fig. 7, where we highlight that line 18 of Fig. 6 is simply replaced by lines 19-22 of Fig. 7.

Finally, whenever a table rotation has to be performed (recall that this depends on the condition to verify) $dict.center$ is updated simply by $dict.center \leftarrow ni$. Specifically, since condition $C1$ is always verified, GA-LEC performs such center update before returning ci (*e.g.*, before line 30 of Fig. 6), while FA-LEC first increments the frequency counter $f[ni]$ and then verifies condition $C2$ (see Fig. 8 showing the snippet pseudo-code which replaces line 30 of Fig. 6 with lines 31-35 to implement the table rotation upon verification of $C2$ in FA-LEC).

```
18: ~~ind ← ni~~
19: ind ← ni − dict.center
20: if ind < 0 then
21:     ind ← ind + dict.T
22: end if
```

Fig. 7. The snippet pseudo-code to make the original EMITCODE function of Fig. 6 adaptive.

```
30: ~~return ci~~
31: f[ni] ← f[ni] + 1
32: if f[ni] ≥ f[dict.center] then
33:     dict.center ← ni
34: end if
35: return ci
```

Fig. 8. The snippet pseudo-code to verify condition C2 of FA-LEC and to accordingly perform the table rotation.

As regards the split versions of the proposed compression schemes (namely, GAS-LEC and FAS-LEC) some other extra logic and structures have to be provided. Just to give some implementation hints, two different dictionary structures $dict1$ and $dict2$ (with their own sizes and pointers to the centers) can be used to produce the group codes a_i. In this case the extra logic to be added would only consist in deciding the actual dictionary structure to use to produce the current codeword: specifically, if $ni < dict1.T$ then the first dictionary structure has to be used, otherwise the second one. Alternatively, the S and L fields of the dictionary structure may be implemented as bi-dimensional arrays, while the centers and the sizes of the sub-tables as arrays of 2 elements. In this case, the $dict.center$ entries would be initialized to 0, while the $dict.T$ ones would be constantly equal to $\lceil T/2 \rceil$ and $\lfloor T/2 \rfloor$. Then, to build a codeword, the modified EMITCODE function would access the right entry of the $dict$ structure by indirection (*e.g.*, $dict.S[p][ind]$, $dict.L[p][ind]$, $dict.center[p]$ and $dict.T[p]$, where p is 0 if $ni < dict.T[0]$ or 1 otherwise). Moreover, in this second case, it would be worth to implement also the frequency counter vector f as a bi-dimensional array, so as to efficiently access its entries, check the rotation condition $C2$ and update the center of the affected sub-table in case of rotation (see lines 24-27 of Fig. 9). Obviously, the most advantageous implementation design depends on the target architecture and is out of the scope of this paper. For the sake of the completeness Fig. 9 shows the pseudo-code of the modified EMITCODE function (namely, EMITCODESPLIT) responsible for producing the codeword ci and rotating the split dictionary (implemented with bi-dimensional fields) upon verification of condition C2 (*e.g.*, FAS-LEC). Finally, it is worth to recall that for GAS-LEC, since condition $C1$ is always verified, a sub-table rotation corresponds to execute only line 26 of Fig. 9.

```
1: function EMITCODESPLIT(di, ni)
2:     if ni < dict.T[0] then
3:         p ← 0
4:         gp ← ni
5:     else
6:         p ← 1
7:         gp ← ni − dict.T[0]
8:     end if
9:     ind ← gp − dict.center[p]
10:    if ind < 0 then
11:        ind ← ind + dict.T[p]
12:    end if
13:    ci.l ← ni + dict.L[p][ind]
14:    if ni == 0 then
15:        ci.r ← dict.S[p][ind]
16:    else
17:        if di < 0 then
18:            di ← di − 1
19:        end if
20:        si ← dict.S[p][ind]
21:        ai ← (di << (32 − ni)) >> dict.L[p][ind]
22:        ci.r ← si|ai
23:    end if
                                ▷ verification of condition C2
24:    f[p][gp] ← f[p][gp] + 1
25:    if f[p][gp] ≥ f[p][dict.center[p]] then
26:        dict.center[p] = gp          ▷ sub-table rotation
27:    end if
28:    return ci
29: end function
```

Fig. 9. The pseudo-code of the function used by GAS-LEC and FAS-LEC to produce the compressed version of the input difference di as a codeword structure ci.

V. EXPERIMENTAL RESULTS

In this section we will assess the performances of the proposed adaptation mechanisms, and compare them against the standard LEC. To this aim, first, we will use various public domain datasets collected by real sensor nodes (Sec. V-B). In order to simplify the discussion, we will focus only on the results obtained by using the split versions of the adaptive schemes and compare them with those obtained by the standard LEC. Indeed, the final goal of this analysis is to prove, by means of a statistical test, that the proposed adaptive schemes outperform LEC when compressing standard environmental signals characterized by high correlation between consecutive samples. Recalling that the latter represents a necessary condition for the LEC algorithm to efficiently compress the input signal, we can consider this scenario as the best-case for LEC. Then, by artificially reducing the temporal correlation of the consecutive samples, we generate 5 datasets from one of the

environmental signals (Sec. V-C). We aim to show that, as the temporal correlation of the signal decreases, the absolute values of the differences between consecutive samples increase and the compression performances of the LEC algorithm decrease. Indeed, the main assumption of the LEC algorithm does not hold anymore since the most probable value of the differences can be different from 0 and the distribution of the differences tends to be more flat. On the other hand, for the proposed adaptive schemes, this situation is less dramatic, since the prefix table is adapted to the trend of the differences between consecutive values. This will result in an increase of the difference between the efficiency obtained by LEC and the efficiency achieved by each proposed adaptive scheme on the five artificial datasets.

A. Performance metric

To evaluate and compare the performances of the different compression algorithms, we will use the notion of efficiency η, defined as the ratio (in percentage) between the information entropy of the differentiated input signal, H_D, and the average number $\overline{bs}$ of bits needed by the algorithm to represent a compressed input difference:

$$\eta = \frac{H_D}{\overline{bs}} \times 100. \tag{5}$$

More in detail, H_D is computed as

$$H_D = -\sum_{j=1}^{J} p(d_j) \cdot \log_2(p(d_j)), \tag{6}$$

where J is the number of possible values of the differences between two consecutive samples and $p(d_j)$ is the probability mass function of difference d_j (we naturally assume that $0 \cdot \log_2(0) = 0$, which is consistent with the well-known limit $\lim_{p \to 0^+} 0 \cdot \log_2(0) = 0$ [26]). Hence, H_D quantifies the average number of bits that an optimal (in the entropy meaning) compression algorithm would nominally use to compress the differentiated input signal. We recall that, generally, the entropy of a monitored signal H is greater than H_D and the strategy of differentiating the original signal is an integral part of the proposed algorithms to easily remove some information redundancy directly at the source.

As regards $\overline{bs}$, it is computed as:

$$\overline{bs} = \frac{1}{N} \cdot \sum_{i=1}^{N} len(bs_i) \tag{7}$$

where N represents the number of samples which compose the signal. Thus, η simply reflects the goodness of a compression algorithm with respect to the ideal entropy compressor.

B. Performance analysis: real environmental signals

To show the effectiveness and validity of GAS-LEC and FAS-LEC, we tested them on various real-world datasets and compared their results, in terms of efficiency, with the ones achieved by LEC. In particular, we used air and surface temperatures, relative humidity and solar radiation measurements from five SensorScope deployments [27], [25]: HES-SO FishNet Deployment, Le Genepi Deployment, Grand-St-Bernard Deployment, PDG 2008 Deployment and Plaine Morte Deployment. We chose to adopt public domain datasets rather than to generate data by ourselves to make the comparison among the compression algorithms as fair as possible. In the following, the five deployments will be referred to by using their corresponding short names, which are shown in Table V. The table also summarizes the number of sensor stations composing each deployment and the corresponding sensor IDs. Finally, since solar radiation and surface temperature measurements are not available for some sensor stations, we marked these stations with superscripted asterisk and minus symbols, respectively.

The WSNs adopted in the deployments employ TinyNode nodes [28], which use a Texas Instruments MSP430 microcontroller [29], a Xemics XE1205 radio transceiver [30], a Sensirion SHT75 sensor module for sensing air temperature and relative humidity [31], a Zytemp TN901 infrared module for sensing surface temperature [32] and a Davis module for sensing solar radiation [33]. Table VI shows the main specifications of the digital sensors employed by the sensor stations for the different measurements (*i.e.*, sensing ranges, expressed in measurement units, and resolutions R of the digitalized samples, expressed in number of bits). We observe that the digital sensor sampling of the air temperature, the surface temperature, the solar radiation and the relative humidity produce digitalized samples with a resolution R of 14, 16, 12 and 12 bits, respectively. Hence, the number of prefix-free codes is 15, 17, 13 and 13, respectively (*i.e.*, $R+1$). It is worth to recall that, while all the tested compression algorithms work on raw data, the datasets corresponding to the five deployments store data in measurement units (*i.e.*, ^oC for air and surface temperatures, $\%$ for relative humidity and W/m^2 for solar radiation). Thus, before applying the algorithms, we extracted the raw data[4] by using the inverted versions of the conversion functions provided in [25].

Fig. 10 shows the values of efficiency of LEC, GAS-LEC and FAS-LEC on all the datasets of the five deployments, grouped by class of measurements. In particular, Fig. 10(a), Fig. 10(b), Fig. 10(c) and Fig. 10(d) visually compare the values of efficiency achieved by the three approaches for air temperature, surface temperature, relative humidity and solar radiation, respectively. The horizontal axes are partitioned into five disjoint segments by using two-headed arrows: each segment represents a different deployment, specified by the corresponding short name.

By analyzing Fig. 10, we can appreciate how the adaptation blocks exploited by GAS-LEC and FAS-LEC allow improving the values of efficiency with respect to LEC. Fig. 11 shows the boxplot diagram of the efficiency obtained by the three algorithms on the four different types of data. For each boxplot, the lowest and the largest efficiency values are represented as whiskers, the lower and upper quartiles are shown with a box, the median is represented by a line, the mean value by a black circle and efficiency values that may be considered outliers are possibly marked with asterisks [34]. To ease a quantitative comparison, Table VII shows the values of the medians of the efficiency distributions obtained by the tested algorithms on the four different types of data. Although the datasets

[4]NaN values were discarded.

TABLE V
THE FIVE SENSORSCOPE DEPLOYMENTS: SUPERSCRIPTED ASTERISK AND MINUS SYMBOLS IDENTIFY THOSE STATIONS FOR WHICH SOLAR RADIATION AND SURFACE TEMPERATURE MEASUREMENTS ARE NOT AVAILABLE, RESPECTIVELY.

Deployment name	Short name	No. stations	IDs
HES-SO FishNet	FN	6	101, 102, 103, 104*, 105*, 106*.
Le Genepi	GE	16	2, 3, 4, 6, 7*, 8, 10*, 11*, 12*, 13*, 15*, 16*, 17, 18*, 19*, 20*.
Grand-St-Bernard	GSB	23	2, 3, 4, 5, 6*, 7*, 8*, 9*, 10*, 11*, 12*, 13*, 14*, 15*, 17*, 18*, 19*, 20*, 25, 28, 29, 31, 32.
PDG 2008	PDG	10	1, 3, 4, 5, 7, 9$^-$, 10, 15, 16, 18.
Plaine Morte	PM	13	2*, 6*, 20, 38*, 58, 74, 78, 83, 90, 91*, 101, 102, 110*.

TABLE VI
MAIN SPECIFICATIONS OF THE DIGITAL SENSORS EMPLOYED BY A TINYNODE NODE FOR COLLECTING DIFFERENT TYPES OF ENVIRONMENTAL MEASURES.

measurement type	sensor module	sensing range [measurement units]	resolution R [bits]
Air temperature	Sensirion SHT75 [31]	$[-20, 60]^oC$	14
Surface temperature	Zytemp TN901 [32]	$[-33, 220]^oC$	16
Air relative humidity	Sensirion SHT75 [31]	$[0, 100]\%$	12
Solar radiation	Davis Solar Radiation [33]	$[0, 1800]W/m^2$	12

TABLE VII
VALUES OF THE MEDIANS OF THE EFFICIENCY DISTRIBUTIONS OBTAINED BY LEC, FAS-LEC AND GAS-LEC ON THE AIR TEMPERATURE, SURFACE TEMPERATURE, RELATIVE HUMIDITY AND SOLAR RADIATION DATASETS OF SENSORSCOPE DEPLOYMENTS.

	LEC	GAS-LEC	FAS-LEC
Air temperature	92.33	95.78	95.22
Surface temperature	87.57	88.57	89.78
Relative humidity	90.84	93.62	93.76
Solar radiation	94.82	97.17	95.14

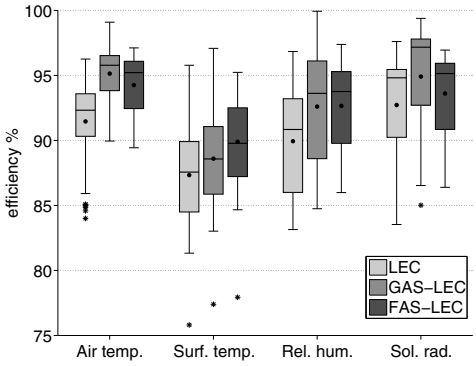
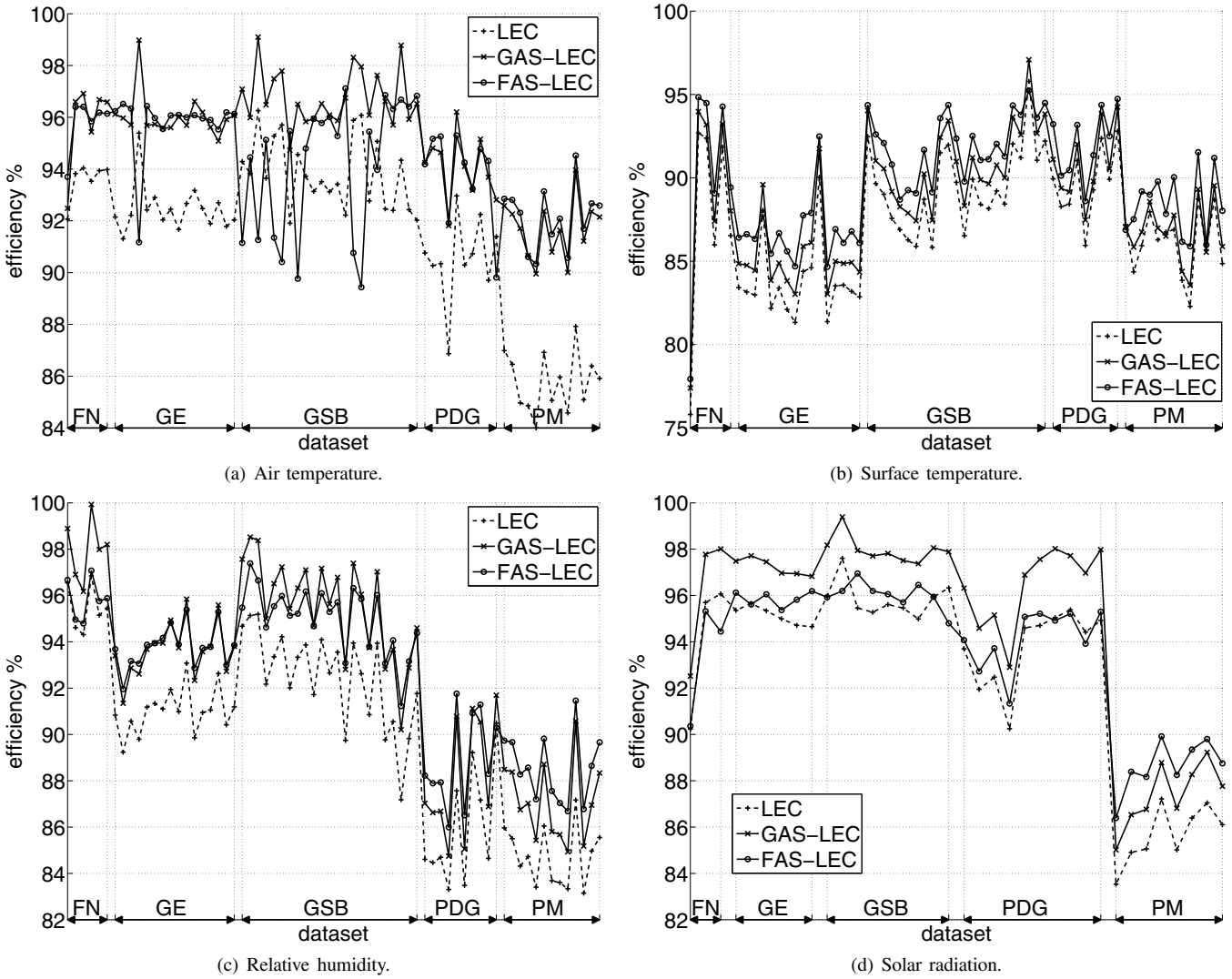


Fig. 11. Boxplots of the values of efficiency obtained by LEC, FAS-LEC and GAS-LEC on the four different types of datasets.

of this experiment represent real environmental signals with rather high correlations between consecutive samples (the ideal situation for the LEC algorithm), we can observe that both the adaptive schemes outperform the LEC compression algorithm.

To verify this impression, we perform a statistical analysis. For each of the three approaches, we merge all the efficiency values achieved on all the datasets by the approach, thus obtaining three distributions. Then, we apply the Friedman test in order to compute a ranking among such distributions [35], and the Iman and Davenport test [36] to evaluate whether there exist statistically relevant differences among them. If there exists a statistical difference, we apply a post-hoc procedure, namely the pairwise Holm test [37]. This test allows detecting effective statistical differences between the pairs of distributions.

Table VIII shows the results of the non-parametric statistical tests: for each algorithm, we show the Friedman rank and the Iman and Davenport *p–value*. If the *p–value* is lower than the level of significance α (in the experiments $\alpha = 0.05$), we can reject the null hypothesis and affirm that there exist statistical differences between the multiple distributions associated with each approach. Otherwise, no statistical difference exists among the distributions and therefore the three distributions are statistically equivalent. We observe that the Iman and Davenport's statistical hypothesis of equivalence is rejected and so statistical differences among the distributions are detected. Thus, we have to apply the pairwise Holm post-hoc procedure. We realize that the statistical hypothesis of equivalence is rejected both between LEC and GAS-LEC, and between LEC and FAS-LEC, while cannot be rejected between GAS-LEC and FAS-LEC. Thus, we can conclude that both the split adaptation blocks proposed in this paper outperform LEC in terms of efficiency. As regards GAS-LEC and FAS-LEC, they are statistically equivalent. We can draw the following conclusions. On the one side, independently of the specific rotation condition, table rotations are beneficial from the compression efficiency point of view. In fact, the two adaptive schemes perform better than the LEC counterpart. On the other side, it can be observed that the early rotations performed by GAS-LEC can be more or less advantageous than the late rotations performed by the FAS-LEC counterpart, and this is confirmed by the statistical analysis: the light (either positive or negative, depending on the specific signal to compress) differences in terms of performances between the two algorithms are likely due to chance.



(a) Air temperature.

(b) Surface temperature.

(c) Relative humidity.

(d) Solar radiation.

Fig. 10. Values of efficiency for LEC, FAS-LEC and GAS-LEC on the (a) air temperature, (b) surface temperature, (c) relative humidity and (d) solar radiation datasets of SensorScope deployments.

C. Performance analysis: weakly correlated signals

LEC is particularly effective when the values of the differences between consecutive samples are close to $d = 0$. In the case of the adaptive schemes, this constraint is less stringent. Indeed, the value of d can be different from 0 and can change dynamically depending on the trend of the signal. In this section, we highlight this feature by compressing less correlated signals. To this aim, we modified the original real datasets by applying a downsampling process at different (downsampling) factors [26]. We had already adopted a similar procedure in [38].

More in detail, we select the portion of the air temperature signal sampled at node 1 of the PDG deployment during the time interval between 16 and 20 April 2008, for a total of 5 monitoring days[5]. Then, such signal is downsampled by 2, 3, 4 and 5 to obtain four different versions of the original dataset sampled every 4, 6, 8 and 10 minutes, respectively. Finally, a variable downsampling procedure is applied to the same signal to obtain a version of the original signal, sampled at variable frequency. Specifically, the portion of the signal corresponding to the first day is kept as-is. The portions of the signal corresponding to the second, third, fourth and fifth days are downsampled by 2 (1 sample each 4 minutes), 3 (1 sample each 6 minutes), 4 (1 sample each 8 minutes), and 5 (1 sample each 10 minutes), respectively. The five signals are then compressed using all the proposed adaptive schemes and the original LEC algorithm.

The aim of this experiment is to evaluate the benefits of the proposed adaptation schemes when compressing environmental signals whose temporal correlation between consecutive samples is less prominent (and not even constant for the variable downsampled signal). Further, it is worth to observe that the constant downsampled signals realistically model those situations in which different sensor devices are deployed in the same zone to monitor the same environmental phenomenon, but at different sampling rates. Finally, the variable downsampled signal models the specific monitoring situation in which a node starts to collect samples at full sampling rate and then progressively decreases the rate as its battery is

[5]We chose this specific dataset because we had verified that its nominal sampling frequency of 1 sample each 2 minutes is accurately respected.

TABLE VIII
RESULTS OF THE NON-PARAMETRIC STATISTICAL TESTS WITH $\alpha = 0.05$.

Friedman test					
Algorithm		Friedman rank	Iman and Davenport *p-value*	Hypothesis	
LEC		2.8912	0	Rejected	
FAS-LEC		1.4770			
GAS-LEC		1.6318			
Pairwise Holm post-hoc procedure					
i	Algorithm	*z-value*	*p-value*	*alpha/i*	Hypothesis
3	LEC vs. FAS-LEC	15.4597	0	0.0167	Rejected
2	LEC vs. GAS-LEC	13.7674	0	0.0250	Rejected
1	FAS-LEC vs. GAS-LEC	1.6923	0.0905	0.0500	Not rejected

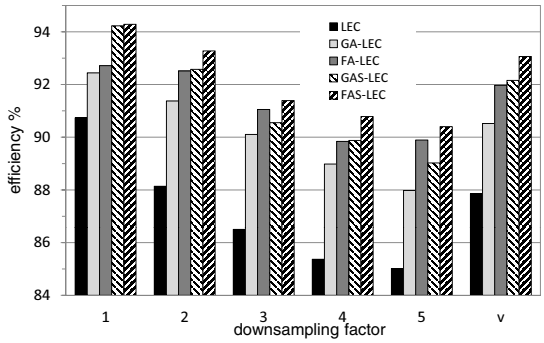


Fig. 12. Values of efficiency obtained by LEC and by all the adaptive compression algorithms on the downsampled datasets.

running low, so as to prolong the whole monitoring application lifetime. It is evident that, in all such circumstances, the option of optimally (and manually) designing different prefix-free tables for the different devices/phenomena/sampling frequencies is not viable, while the proposed adaptive mechanisms help reducing the effects of not-optimally compressing the corresponding signals.

Fig. 12 shows the efficiency of the proposed adaptive schemes compared with the LEC algorithm, when compressing the different downsampled signals (v denotes the signal generated by the variable downsampling procedure). For the sake of completeness, the efficiency of the tested algorithms when compressing the original signal (*e.g.*, downsampling factor equal to 1) are also shown in the figure (first group of bars).

By analyzing Fig. 12 we can observe that, though the decreased temporal correlations of the input signals affect the efficiencies of all the algorithms, the proposed adaptive schemes are more effective than the original LEC algorithm. In particular, we note that the values of efficiency of the split versions (GAS-LEC and FAS-LEC) are higher than the values of the corresponding non-split versions (GA-LEC and FA-LEC). Further, the adaptive schemes based on condition C2 (FA-LEC and FAS-LEC) outperform those based on condition C1 (GA-LEC and GAS-LEC). For example, the efficiency of the LEC algorithm decays from 90.7% of the original signal to 85.0% when the signal is downsampled by 5, while the efficiency of FAS-LEC decays from 94.3% to 90.4%.

VI. CONCLUSIONS

In this paper, we have proposed two simple adaptation mechanisms for supplying the lossless compression algorithm (LEC), previously proposed by us in the context of wireless sensor networks, with the property of adapting itself to different types of signals, though preserving its simplicity and its suitability for the limited resources available on–board sensor nodes. In particular, both the mechanisms rely on the novel concept of appropriately rotating the prefix-free tables upon certain conditions. The proposed schemes were tested on several datasets collected by real sensor network deployments. The results show that the adaptation mechanisms allow improving the compression efficiency of LEC. Since compression permits to reduce the amount of data transmitted/received by a sensor node, this improvement in terms of compression efficiency allows saving energy and therefore extending the lifetime of the sensor nodes.

ACKNOWLEDGMENT

This work was supported by the EU Integrated Project iCore, *"Internet Connected Objects for Reconfigurable Ecosystems"* (http://www.iot-icore.eu/), funded within the European 7th Framework Programme (contract number: 287708).

REFERENCES

[1] A. Gavras, A. Karila, S. Fdida, M. May, and M. Potts, "Future Internet research and experimentation: the FIRE initiative," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 3, pp. 89–92, July 2007.
[2] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: a survey," *Comput. Netw.*, vol. 54, no. 15, pp. 2787–2805, 2010.
[3] O. Vermesan, P. Friess, P. Guillemin, S. Gusmeroli, H. Sundmaeker, A. Bassi, I. S. Jubert, M. Mazura, M. Harrison, M. Eisenhauer, and P. Doody, "Internet of Things strategic research agenda," in *Internet of Things: Global Technological and Societal Trends*, O. Vermensan and P. Friess, Eds. River Publishers, 2011.
[4] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of Things: vision, applications and research challenges," *Ad Hoc Netw.*, vol. 10, no. 7, pp. 1497–1516, 2012.
[5] "Forrester research homepage," 2012. Available: http://www.forrester.com/
[6] "Controls drives & automation," Aug. 2012. Available: http://content.yudu.com/Library/A1ykj7/ControlsDrivesampAut/resources/44.htm
[7] O. Vermesan, P. Friess, G. Woysch, P. Guillemin, S. Gusmeroli, H. Sundmaeker, A. Bassi, M. Eisenhauer, and K. Moessner, "Europe's IoT strategic research agenda 2012," in *The Internet of Things 2012 - New Horizons*, O. Vermesan, P. Friess, and A. Furness, Eds. Halifax, UK, 2012.
[8] C. Alcaraz, P. Najera, J. Lopez, and R. Roman, "Wireless sensor networks and the Internet of Things: do we need a complete integration?" in *Proc. 2010 International Workshop Security Internet Things.*
[9] (2013) Libelium homepage. Available: http://www.libelium.com/
[10] (2013) Cosm homepage. Available: http://cosm.com/
[11] F. Marcelloni and M. Vecchio, "An efficient lossless compression algorithm for tiny nodes of monitoring wireless sensor networks," *Comput. J.*, vol. 52, no. 8, pp. 969–987, 2009.

[12] F. Marcelloni and M. Vecchio, "A simple algorithm for data compression in wireless sensor networks," *IEEE Commun. Lett.*, vol. 12, no. 6, pp. 411–413, June 2008.
[13] T. Srisooksai, K. Keamarungsi, P. Lamsrichan, and K. Araki, "Practical data compression in wireless sensor networks: a survey," *J. Netw. Comput. Appl.*, vol. 35, no. 1, pp. 37–59, 2012.
[14] S. W. Golomb, "Run-length encodings," *IEEE Trans. Inf. Theory*, vol. 12, pp. 399–401, Sept. 1966.
[15] P. Elias, "Universal codeword sets and representations of the integers," *IEEE Trans. Inf. Theory*, vol. 21, no. 2, pp. 194–203, Mar 1975.
[16] G. Chrysos and I. Papaefstathiou, "Heavily reducing WSNs' energy consumption by employing hardware-based compression," in *Ad-Hoc, Mobile Wireless Netw.*, ser. Lect. Notes Comput. Sc., P. Ruiz and J. Garcia-Luna-Aceves, Eds. Springer Berlin Heidelberg, 2009, vol. 5793, pp. 312–326.
[17] (2013) Xilinx homepage. Available: http://www.xilinx.com/
[18] (2013) Crossbow Technology homepage. Available: http://bullseye.xbow.com:81/
[19] A. K. Maurya, D. Singh, and A. K. Sarje, "Median predictor based data compression algorithm for wireless sensor network," *International J. Smart Sensors Ad Hoc Netw.*, vol. 1, no. 1, pp. 62–65, 2011.
[20] J. G. Kolo, S. A. Shanmugam, D. W. G. Lim, L. M. Ang, and K. P. Seng, "An adaptive lossless data compression scheme for wireless sensor networks," *J. Sensors*, 2012.
[21] C. Tharini and P. V. Ranjan, "Design of modified adaptive Huffman data compression algorithm for wireless sensor network," *J. Comput. Science*, vol. 5, no. 6, pp. 466–460, 2009.
[22] J. S. Vitter, "Design and analysis of dynamic Huffman codes," *J. ACM*, vol. 34, no. 4, pp. 825–845, Oct. 1987.
[23] J. S. Vitter, "Algorithm 673: dynamic Huffman coding," *ACM Trans. Math. Softw.*, vol. 15, no. 2, pp. 158–167, June 1989.
[24] K. C. Barr and K. Asanović, "Energy-aware lossless data compression," *ACM Trans. Comput. Syst.*, vol. 24, no. 3, pp. 250–291, Aug. 2006.
[25] (2013) Sensorscope homepage. Available: http://lcav.epfl.ch/sensorscope-en/
[26] D. Salomon, *Data Compression: The Complete Reference*, 4th ed. Springer-Verlag, 2007.
[27] F. Ingelrest, G. Barrenetxea, G. Schaefer, M. Vetterli, O. Couach, and M. Parlange, "Sensorscope: application-specific sensor network for environmental monitoring," *ACM Trans. Sen. Netw.*, vol. 6, no. 2, pp. 17:1–17:32, Mar. 2010.
[28] H. D. Ferrière, L. Fabre, R. Meier, and P. Metrailler, "TinyNode: a comprehensive platform for wireless sensor network applications," in *Proc. 2006 International Conf. Inf. Process. Sensor Netw.*, pp. 358–365.
[29] (2013) Texas Instruments homepage. Available: http://www.ti.com/
[30] (2013) Xemics homepage. Available: http://www.xemics.com/
[31] (2013) Sensirion homepage. Available: http://www.sensirion.com/
[32] (2013) Zytemp homepage. Available: http://www.zytemp.com/
[33] (2013) Davis homepage. Available: http://www.davisnet.com/
[34] J. W. Tukey, *Exploratory Data Analysis*. Addison-Wesley, 1977.
[35] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *J. Amer. Statist. Assoc.*, vol. 32, no. 200, pp. 675-701, 1937.
[36] R. L. Iman and J. H. Davenport, "Approximations of the critical region of the Friedman statistic," *Commun. Stat. A-Theor.*, vol. 9, pp. 571-595, 1980.
[37] S. Holm, "A simple sequentially rejective multiple test procedure," *Scand. J. Stat.*, vol. 6, no. 2, pp. 65-70, 1979.
[38] F. Marcelloni and M. Vecchio, "Enabling compression in tiny wireless sensor nodes," in *Wireless Sensor Networks*, S. Tarannum, Ed. InTech, 2011, ch. 12, pp. 257–276.

Massimo Vecchio received the Laurea degree in Computer Engineering (Magna cum Laude) from the University of Pisa and the Ph.D. degree in Computer Science and Engineering (with Doctor Europaeus mention) from IMT Lucca Institute for Advanced Studies in 2005 and 2009, respectively. His research background is on computational and artificial intelligence techniques, such as metaheuristics for global optimization and fuzzy logic. During his Ph.D. degree, however, his research interests moved towards power-efficient engineering and application designs for pervasive systems and devices. From October 2008 to March 2010 he worked as simulation engineer at INRIA-Saclay (France). Then, he joined the Signal Processing in Communications group at the University of Vigo (Spain) as a post-doctoral researcher, until September 2012. Recently he moved back to Italy to work as senior researcher at CREATE-NET, mainly in the field of Internet of Things devices and resources virtualization. Besides, current Massimo's research interests include metaheuristic and stochastic methods for wireless sensor nodes self-localization, node mobility for efficient data collection and power-aware consensus techniques. He is author of one book monograph and co-author of two book chapters, more than 10 journal papers and various conference papers.

Raffaele Giaffreda graduated at Politecnico Torino (Italy, 1995) and also holds an MSc in Telecom Engineering from University College of London (UK, 2001). He has a wide technology background in the telecommunications domain, spanning from optical backhaul networks to wireless access ones with research emphasis on dynamic resource usage optimisation, information retrieval and context awareness. He worked in the research departments of Telecom Italia (1994-1995, self-healing optical communication rings), British Telecom (1998-2008, information retrieval systems, context-aware networks and service delivery platforms) and is currently the head of the research group smaRt IOT (RIOT) at CREATE-NET, focusing on cognitive Internet of Things and associated proof-of-concept implementation and related applications. He has been involved in many international collaborative projects and is currently the coordinator of the EU FP7 Internet of Things project iCore (www.iot-icore.eu). He has more than 30 co-authored publications in international journals, conferences and workshops.

Francesco Marcelloni received the Laurea degree in Electronics Engineering and the Ph.D. degree in Computer Engineering from the University of Pisa in 1991 and 1996, respectively. He is currently an associate professor at the University of Pisa. He has co-founded the Computational Intelligence Group at the Department of Information Engineering of the University of Pisa in 2002. Further, he is the founder and head of the Competence Centre on MObile Value Added Services (MOVAS). His main research interests include multi-objective evolutionary algorithms, genetic fuzzy systems, fuzzy clustering algorithms, pattern recognition, signal analysis, neural networks, mobile information systems, and data compression and aggregation in wireless sensor networks. He has co-edited three volumes, four journal special issues, and is (co-)author of a book and of more than 180 papers in international journals, books and conference proceedings. He has been TPC co-chair of the 9th International Conference on Intelligent Systems Design and Applications (ISDA'09) and ISDA'11, TPC chair of the 8th IEEE International workshop on Sensor Networks and Systems for Pervasive Computing, general co-chair of ISDA'10, and co-organizer of several workshops and special sessions in international conferences. Currently, he serves as associate editor of *Information Sciences* (Elsevier) and *Soft Computing* (Springer) and is on the editorial board of other four international journals.