Information Fusion 52 (2019) 13–30

Contents lists available at ScienceDirect

Information Fusion

journal homepage: www.elsevier.com/locate/inffus

Full Length Article

Short-long term anomaly detection in wireless sensor networks based on machine learning and multi-parameterized edit distance

Francesco Cauteruccio[a], Giancarlo Fortino[a,b], Antonio Guerrieri[b], Antonio Liotta[c], Decebal Constantin Mocanu[d], Cristian Perra[e], Giorgio Terracina[a,*], Maria Torres Vega[f]

[a] University of Calabria, Italy
[b] ICAR-CNR, Italy
[c] University of Derby, UK
[d] Technical University of Eindhoven, the Netherlands
[e] University of Cagliari, Italy
[f] Ghent University, Belgium

ARTICLE INFO

Keywords:
Intelligent sensing
Sensor fusion
Anomaly detection
Cloud-assisted sensing
Internet of Things

ABSTRACT

Heterogeneous wireless sensor networks are a source of large amount of different information representing environmental aspects such as light, temperature, and humidity. A very important research problem related to the analysis of the sensor data is the detection of relevant anomalies. In this work, we focus on the detection of unexpected sensor data resulting either from the sensor system itself or from the environment under scrutiny. We propose a novel approach for automatic anomaly detection in heterogeneous sensor networks based on coupling edge data analysis with cloud data analysis. The former exploits a fully unsupervised artificial neural network algorithm, whereas cloud data analysis exploits the multi-parameterized edit distance algorithm. The experimental evaluation of the proposed method is performed applying the edge and cloud analysis on real data that has been acquired in an indoor building environment and then distorted with a range of synthetic impairments. The obtained results show that the proposed method can self-adapt to the environment variations and correctly identify the anomalies. We show how the combination of edge and cloud computing can mitigate the drawbacks of purely edge-based analysis or purely cloud-based solutions.

1. Introduction

A wireless sensor network (WSN) is a distributed network architecture composed of a set of autonomous networked electronic devices (sensor nodes) collecting data from the surrounding environment. Examples of data sources are temperature, humidity, light, noise, electric current, voltage, and power.

The market of wireless sensor networks is continuously growing thanks to technological and computational improvements [1]. At the same time, efficient management techniques are needed for dealing with the network complexity and the huge amount and variety of sensor data [2,3].

Wireless sensor networks are typically connected to cloud services through the Internet. Cloud platforms provide the storage and computing infrastructures necessary for archiving and processing the large amount of data generated by sensors [4]. Graphical visualization, statistical analysis, and tabular reporting of sensor data are very common applications in WSNs and in the Internet of Things (IoT) domain.

A challenging research is the problem of sensor data analysis for automatic anomaly detection [5]. The term anomaly detection has a broad meaning in the literature, referring to the identification of items, events or observations which rise some kind of suspicions. In this paper, we focus on the detection of unexpected variations of sensed data that may result from the sensor system itself but also from the environment under scrutiny. In WSNs, the causes of anomalies may be related to several factors. Examples are: devices running out of power, devices deviating from the expected behaviour, and malfunctioning devices. Yet, it is often difficult to discern the anomalies of the sensor system from the actual anomalies in the sensed environment.

In this context, the kind of WSN, the detection methodology, and the kind of anomalies of interest may significantly impact on the solution design. In this paper, we focus on three orthogonal research directions related to anomaly detection in WSNs: *(i)* homogeneous vs. heterogeneous WSNs; *(ii)* methods directly running on sensing devices (hereafter, edge-based methods) vs. methods running on the cloud (hereafter, cloud-based methods); *(iii)* anomalies spanning over short periods of

* Corresponding author.
E-mail address: terracina@mat.unical.it (G. Terracina).

https://doi.org/10.1016/j.inffus.2018.11.010
Received 13 June 2018; Received in revised form 7 November 2018; Accepted 25 November 2018
Available online 26 November 2018

time (hereafter, short-term anomalies) vs. anomalies spanning over long periods of time (hereafter, long-term anomalies).

Anomaly detection in homogeneous WSNs received much attention in the literature. Most of the approaches regarding anomaly detection are dedicated to the analysis of data streams produced by a *single* device [6–9]. In this case, a single device is analyzed, by means of different techniques, to understand whether or not an anomaly has occurred. These techniques are usually based on complex mathematical analysis or statistical methods applied on data streams [6,9], which are tailored to the specific numerical characteristics of the kind of sensed data. Consequently, applying such methods to heterogeneous WSNs to sense different kind of parameters, and involving multiple sensors is not straightforward.

Data representations other than the numerical ones have been considered in [10–17] which, however, assume that the actual data is homogeneous. For instance, in [10] a survey on graph-based anomaly detection and description is presented. Its focus is on providing a general and structured overview of methods for anomaly detection in data represented as graphs and categorized under various settings. Being able to differentiate data representation allows to apply anomaly detection in different domains such as financial auctioning and social networks. In particular, anomaly detection on (or based on) social network has gained an increasing importance [11]. Other approaches apply mathematical or machine learning based analysis on different data levels. This kind of techniques has been applied in intrusion detection in security systems [12] and fraud detection for credit cards [13]. In [14], incoming data packets are compared to fixed patterns in order to identify known behavioral instances. Spatial anomaly detection is analyzed in [15] using neural networks.

Even in the presence of numerical data only, a sensor network may be heterogeneous, if it consists of sensor nodes with different abilities. Heterogeneous sensors are devices producing different kinds of signals, measures or messages. As an example, sensors in an heterogeneous network may produce differently scaled real value data, measuring different parameters like temperature, humidity, light, electric current, voltage, and so on.

Anomaly detection in heterogeneous sensor networks has received less attention in the literature. An approach for monitoring heterogeneous wireless sensor networks and to identify hidden correlations between heterogeneous sensors has been proposed in [18]. This approach can identify hidden correlations between heterogeneous sensors but has not been specifically conceived for anomaly detection. Moreover, this method has not been designed for large sensor networks, and would be unfeasible in this context, given its requirement to make comparisons between all the sensor pairs across the network. Furthermore, it would be unrealistic to rely on the entire data stream, given the limited resources available in the network.

In this paper we develop a framework to detect anomalies in *heterogeneous* WSNs. The proposed framework combines two different approaches: the first one locally analyzes the sensor data coming from the individual nodes in the network (each node may contain heterogeneous sensors); the second one compares data coming from several heterogeneous sensors spread over the entire network. We show that the combined use of local and cloud-based analysis allows to overcome the limitations arising when each method is used in isolation, allowing us to detect more complex anomalies and to operate on a larger WSN scale.

As far as cloud-based methods and edge-based methods are concerned, we point out that performing anomaly detection on the cloud allows to resort to quite complex algorithms and, consequently, to get accurate results. However, performing anomaly detection just on the cloud presents some drawbacks. First of all, communication bottlenecks yielded by too much data transmitted from nodes to the central servers may induce packet losses and delays [19]. Moreover, in large WSNs simultaneously analyzing all sensors streams would introduce real-time computational constraints.

On the other hand, edge-based methods run directly on nodes equipped with light computing power. Most of these approaches require samples of historical data to be kept in the sensor node, which has limited memory. Besides that, most of the state-of-the-art learning algorithms target network organization, usually routing protocols [19]. Yet, few works target directly in-node anomaly detection, and all methods still depend on relatively large sliding windows for accuracy. For instance, a sliding window is used together with an ellipsoidal support vector machine (SVM) in [20], with various linear and non-linear machine learning models in [21,22], and with ensemble methods [23].

Given the intrinsic nature of edge-based methods, these must be light-weight and fast. However, due to the stringent constraints they must comply with, edge computing cannot be as accurate as cloud-based methods. Thus, edge-based methods would naturally be more scalable than cloud-based ones, but suffer from poor accuracy. We aim to address this weakness in our work.

Our framework combines an edge-based method with a cloud-based one in order to overcome the drawbacks that each method would have when used in isolation. Running locally in each device, the edge-based method acts in real-time on the sensor data, providing the first line of the anomaly detection process. Edge detection does not aim at high accuracy; it is intended to prompt the cloud system towards analyzing a subset of sensor streams, as we describe in Section 2.

Using machine learning (ML) on the device poses a new problem: how to detect anomalies under constrained computing conditions. We introduce a novel ML approach, named *Anomaly detection with Generative replay* (AnGe). AnGe can detect anomalies, by making use of the generative and density estimation capabilities of a deep learning method, i.e. restricted Boltzmann machines, without requiring to store any historical data in the node memory. On the other hand, the proposed cloud-based method extends the work presented in [18], in order to identify anomalies. Computational costs are reduced both by a redesign of the approach and by the preemptive action of the edge algorithm, which selects only the most probable sensor streams for analysis.

Another important aspect to consider is the time-span covered by the anomaly itself. Due to the particular constraints of edge and cloud computing, we should not try to detect both short- and long-term anomalies at the same time. Edge methods can only rely on limited computing resources, which makes it very hard to detect long-term anomalies directly in the nodes. We focus on short-term anomalies at the edge, leaving long-term anomalies to the cloud. Our approach is therefore particularly effective at detecting a range of anomalies, by tracing the short-term origin of long-term anomalies.

Thus, our contribution goes even beyond the issue of scalability in IoT anomaly detection. We can detect short-term anomalies that would escape a cloud-based system and, conversely, long-term anomalies that would be impossible to capture on board of sensor nodes.

The paper is organized as follows. Section 2 presents the proposed framework, including the cloud-based and the edge-based computing components. The experimental analysis and related discussion are reported in Section 3. Finally, Section 4 draws the conclusions.

2. Proposed framework

In this section we introduce our general framework for the Short-Long Term Anomaly Detection method, which mixes an edge computing based approach for the identification of short-term anomalies, and a cloud computing based approach for the identification of long-term anomalies.

This mixed approach aims at mitigating the drawbacks that each of these two methods would have when used in isolation, while making the most of individual strengths. The edge-based method does not exploit historical data and, consequently, its hardware and computational requirements are low. Due to these properties, the method is also effective at identifying alterations in the data that occur in a short period of time; however, it may miss variations on sensor data spanning over

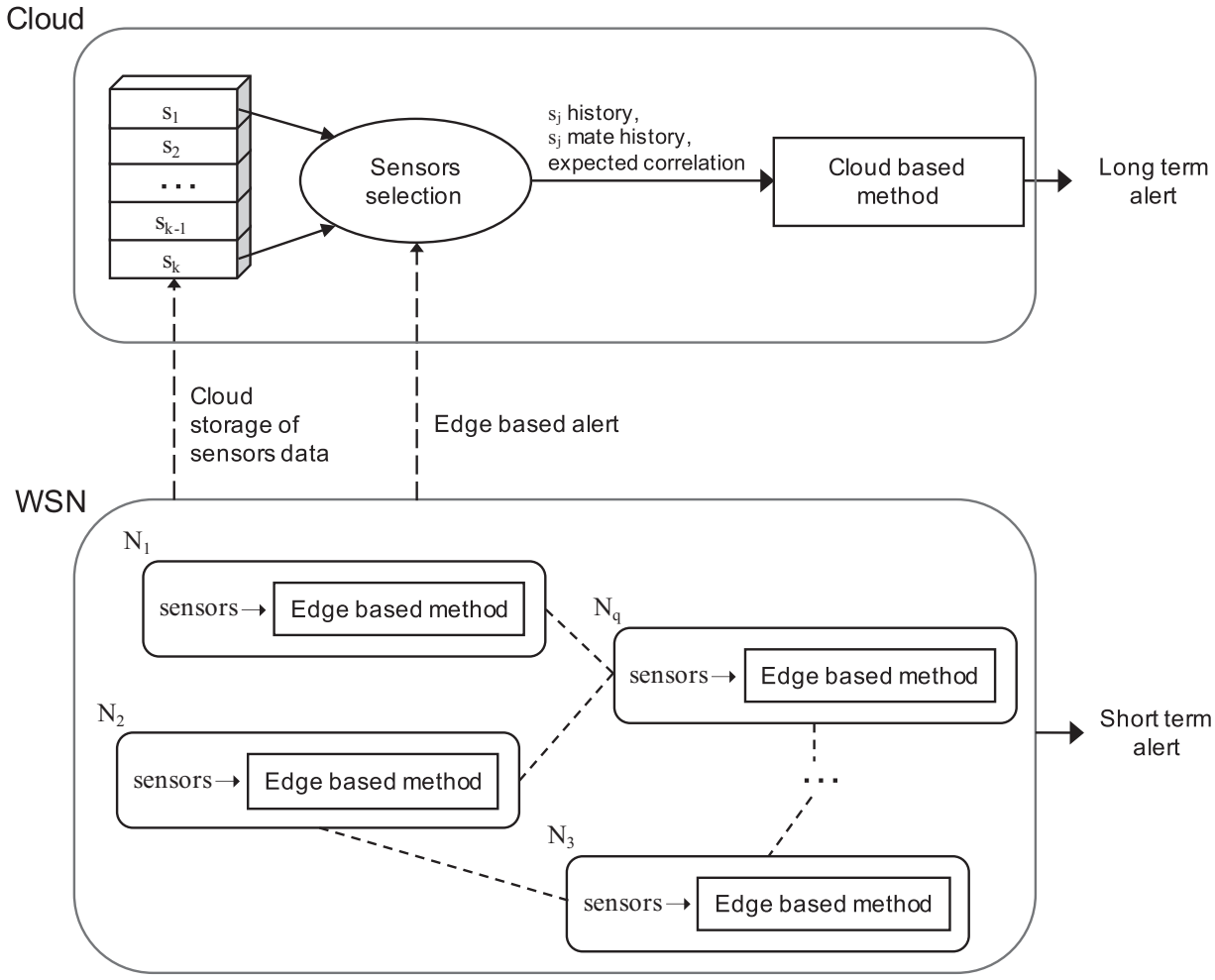


Fig. 1. The main workflow of the proposed approach.

relatively longer periods. If the anomaly is isolated and short termed it may result from some localized problem or may just arise from noise. Per contra, when a local anomaly is at the root of a bigger (long termed) issue on a sensor, a purely edge-based detection may fail.

On the other hand, a cloud-based method tends to be more accurate at identifying long-term anomalies; but it may miss very short ones. However, in order to carry out its task, the method needs to analyze the history of sensed data, which requires significant computational efforts. We cannot afford this effort on all sensor streams, that is where edge processes come handy to help selecting the relevant streams and reduce load on network and cloud.

In the proposed framework, we mix the two methods as shown in Fig. 1. The edge-based method runs continuously on each node of the WSN. No communication between the nodes is needed to perform the analysis. Short-term anomaly detection is carried out at node level. As soon as a short-term anomaly is identified on a node, a short-term alert is issued and sent also to the cloud for further elaboration.

At the cloud level, information on the alerted node is exploited to identify sensors of interest first (see Section 2.3 for the details). Then, an instance of the cloud-based method is immediately triggered, exclusively for each of these sensors. It starts working on a data window already available in the cloud and, consequently, it does not have to wait for data generation; it issues a long-term alert as soon as a long-term anomaly is detected on the previous data window. The task on each sensor is repeatedly run until no long-term anomalies are detected on it for a given period of time; after this, the task is paused until a new short-term alert regarding the sensor is received.

2.1. Preliminaries

Both the edge-based and the cloud-based methods proposed in this paper build on previous work by the authors. For completeness and for the sake of the non-specialist reader, we provide below a summary of key concepts that underlie the framework introduced hereafter.

2.1.1. Preliminary notions for the edge-based method

In order to perform edge-based anomaly detection we contribute by exploiting the possibility of performing online unsupervised learning in each node with Artificial Neural Networks (ANN). This ensures a fully decentralized method to detect anomalies, each node being completely independent from the others. At the same time, it ensures data fusion for one node, i.e. the measurements of all sensors belonging to one node are treated together at each time step t to detect anomalies.

However, online learning with ANNs is in many cases difficult due to the need of storing and relearning large amounts of previous experiences, in order to avoid catastrophic forgetting. While for a standard computer this is an issue that can be easily solved, in the world of low-resources devices these excessive memory requirements, necessary to explicitly store previous observations, represent a big challenge. To overpass it, in this paper, we make use of a novel concept proposed by us in [24] and developed further in [25,26], namely *generative replay*. Generative replay uses the generative capabilities of generative ANN models to generate approximations of past experiences, instead of recording them, as experience replay does. Thus, the generative model can be trained online, and does not require the system to store any of the

observed data points, this being a perfect scenario for anomaly detection in wireless sensor nodes. More exactly, in this paper, we use a generative model called Restricted Boltzmann Machine (RBM) [27] trained with *Online Contrastive Divergence with Generative Replay* (OCD_{GR}), and named RBM_{OCD} [24].

In the approaches described in [24–26], generative replay is capable just to learn data distributions in an online manner, but it cannot perform online anomaly detection. In this paper, we address this issues, and we propose a novel method based on RBM_{OCD} and generative replay to perform online anomaly detection. In the next paragraphs RBM_{OCD} and similarity metrics with RBMs are briefly summarized for the benefit on the non-specialist reader, whereas in Section 2.2, the new proposed method for online anomaly detection is introduced.

RBMs have been introduced in [27] as a powerful model to learn a probability distribution over its inputs. Formally, RBMs are generative stochastic neural networks with two binary layers: the hidden layer $\mathbf{h} = [h_1, h_2, .., h_{n_h}]$, and the visible layer $\mathbf{v} = [v_1, v_2, .., v_{n_v}]$, where n_h and n_v are the numbers of hidden neurons and visible neurons, respectively. In comparison with the original Boltzmann machine [28], the RBM architecture is restricted to be a complete bipartite graph between the hidden and visible layers, disallowing intra-layer connections between the units. The energy function of an RBM for any state $\{\mathbf{v}, \mathbf{h}\}$ is computed by summing over all possible interactions between neurons, weights, and biases as follows:

$$E(\mathbf{v}, \mathbf{h}) = -\mathbf{a}^T\mathbf{v} - \mathbf{b}^T\mathbf{h} - \mathbf{h}^T\mathbf{W}\mathbf{v}, \tag{1}$$

where $\mathbf{W} \in \mathbb{R}^{n_h \times n_v}$ is the weighted adjacency matrix for the bipartite connections between the visible and hidden layers, and $\mathbf{a} \in \mathbb{R}^{n_v}$ and $\mathbf{b} \in \mathbb{R}^{n_h}$ are vectors containing the biases for the visible and hidden neurons, respectively. Functionally, the visible layer encodes the data, while the hidden layer increases the learning capacity of the RBM model by enlarging the class of distributions that can be represented to an arbitrary complexity [29]. The activations of the hidden or visible layers are generated by sampling from a sigmoid $S(\cdot)$ according to: $P(\mathbf{h}) = S(\mathbf{b} + \mathbf{W}\mathbf{v})$ and $P(\mathbf{v}) = S(\mathbf{a} + \mathbf{W}^T h)$.

Motivated by the facts that: (1) hippocampal replay [30] in the human brain does not recall previous observations explicitly, but instead it generates approximate reconstructions of the past experiences for recall, and (2) RBMs can generate good samples of the incorporated data distribution via Gibbs sampling [31], in [24] we proposed RBM_{OCD}. Intuitively, RBM_{OCD} uses generated samples by itself (instead of recalling previous observations from stored memory) during the online training process. Thus, the RBM model can retain knowledge of past observations while learning new ones. The interested reader is referred to [24] for a detailed discussion on RBM_{OCD}.

Just like for any other RBM variant trained offline [32], during learning RBM_{OCD} minimizes the error between the reconstructed version of the input data, denoted further $\hat{\mathbf{v}}$, and the input data itself, $\mathbf{v}$. The reconstructed version ($\hat{\mathbf{x}}$) of a given input data point ($\mathbf{x}$), is computed by performing a one step Gibbs sampling starting from the original data point clamped to the visible neurons ($\mathbf{v}$), then by inferring the hidden neurons activation ($\mathbf{h}$), and then by inferring the visible neurons activation $\hat{\mathbf{v}}$ given the hidden neurons activation. The values of the latter one activation give $\hat{\mathbf{x}}$. Moreover, in our previous work [33], we showed that the error computed between a testing data point and its reconstructed version given by an already trained offline RBM, can be used as a similarity metric. More exactly, it can say how far the testing data point is from the training data distribution. The interested reader is referred to [33–36] for more thorough discussions.

2.1.2. Preliminary notions for the cloud-based method

In order to identify long-term anomalies, we exploit our recently introduced string similarity metric, called Multi-Parameterized Edit Distance (hereafter, MPED) [18], to measure long-term correlations between apparently unrelated data. In fact, given a pair of sensors, MPED is able to identify *hidden* correlations between them even if they measure different kind of events; this will allow us to define a method to detect expected correlations between pairs of sensors first, and to verify expected correlations later on, during the normal operation of the sensors in a network.

MPED allows the computation of the minimum edit distance between two strings, provided that finding the optimal matching schema, under a set of constraints, is part of the problem. In order to understand how MPED works, in the following, we briefly recall the theoretical components of MPED.

First of all, the notion of matching schema must be introduced, which is the core ingredient of MPED.

Let Π_1 and Π_2 be two (possibly disjoint) *alphabets* of symbols and let s_1 and s_2 be two strings defined over Π_1 and Π_2, respectively. A matching schema M over Π_1 and Π_2 is a schema representing how different combinations of the alphabets Π_1 and Π_2 can be combined via matching. Intuitively, given two strings s_1 and s_2 defined over Π_1 and Π_2, M states which symbols of s_1 can be considered matching with symbols of s_2. Many-to-many matchings are expressed with π-partitions, and partitions disallow ambiguous matchings. The following definitions introduce M formally.

Definition 2.1 (. π-partition) Given an alphabet Π and an integer π such that $0 < \pi \leq |\Pi|$, a *π-partition* is a partition Φ^π of Π such that $0 < |\phi_v| \leq \pi$, for each $\phi_v \in \Phi^\pi$.

Definition 2.2 (. $\langle \pi_1, \pi_2 \rangle$-matching schema) Given two alphabets Π_1 and Π_2 and two integers π_1 and π_2, *a $\langle \pi_1, \pi_2 \rangle$-matching schema* is a function $M_{\langle \pi_1, \pi_2 \rangle} : \Phi_1^{\pi_1} \times \Phi_2^{\pi_2} \rightarrow \{true, false\}$, where $\Phi_i^{\pi_i}$ ($i \in \{1, 2\}$) is a π_i-partition of Π_i and, for each $\phi_v \in \Phi_1^{\pi_1}$ (resp., $\phi_w \in \Phi_2^{\pi_2}$), there is at most one $\phi_w \in \Phi_2^{\pi_2}$ (resp., $\phi_v \in \Phi_1^{\pi_1}$) such that $M(\phi_v, \phi_w) = true$. This means that all the symbols in ϕ_v match with all the ones in ϕ_w. $M(\phi_v, \phi_w) = false$ indicates that all the symbols in ϕ_v mismatch with all the ones in ϕ_w.

Once the notion of matching schema is available, the definition of distance between two strings is formally introduced by the following definitions.

Definition 2.3 (Transposition)**.** Let s_1 and s_2 be two strings defined over the alphabets Π_1 and Π_2. Let - be a symbol not included in $\Pi_1 \cup \Pi_2$. Then, a string $\bar{s}_i$ over $\Pi_i \cup \{-\}$ ($i \in 1, 2$) is a *transposition* of s_i if $\bar{s}_i$ can be obtained from s_i by deleting all the occurrences of - . The set of all the possible transpositions of s_i is denoted by $\mathcal{TR}(s_i)$.

Definition 2.4 (Alignment)**.** An *alignment* for the strings s_1 and s_2 is a pair $\langle \bar{s}_1, \bar{s}_2 \rangle$, where $\bar{s}_1 \in \mathcal{TR}(s_1)$, $\bar{s}_2 \in \mathcal{TR}(s_2)$ and $len(\bar{s}_1) = len(\bar{s}_2)$.

Definition 2.5 (Match and distance)**.** Let $\langle \bar{s}_1, \bar{s}_2 \rangle$ be an alignment for s_1 and s_2, let $M_{\langle \pi_1, \pi_2 \rangle}$ be a $\langle \pi_1, \pi_2 \rangle$-matching schema over π-partitions $\Phi_1^{\pi_1}$ and $\Phi_2^{\pi_2}$, and let j be a position with $1 \leq j \leq len(\bar{s}_1) = len(\bar{s}_2)$. We say that $\langle \bar{s}_1, \bar{s}_2 \rangle$ has a match at j if:

- $s_1[j] \in \phi_v$, $s_2[j] \in \phi_w$, $\phi_v \in \Phi_1^{\pi_1}$, $\phi_w \in \Phi_2^{\pi_2}$ and $M_{\langle \pi_1, \pi_2 \rangle}(\phi_v, \phi_w) = true$.

The *distance* between $\bar{s}_1$ and $\bar{s}_2$ under $M_{\langle \pi_1, \pi_2 \rangle}$ is the number of positions at which the pair $\langle \bar{s}_1, \bar{s}_2 \rangle$ does not have a match.

Given the previous definitions, we can introduce the notion of *Multi-Parameterized Edit Distance* between two strings s_1 and s_2 as follows:

Definition 2.6 (Multi-Parameterized Edit Distance - MPED)**.** Let π_1 and π_2 be two integers such that $0 < \pi_1 \leq |\Pi_2|$ and $0 < \pi_2 \leq |\Pi_1|$; the Multi-Parameterized Edit Distance between s_1 and s_2 ($\mathcal{L}_{\langle \pi_1, \pi_2 \rangle}(s_1, s_2)$, for short) is the minimum distance that can be obtained with any $\langle \pi_1, \pi_2 \rangle$-matching schema and any alignment $\langle \bar{s}_1, \bar{s}_2 \rangle$.

To better understand the given definitions, we next present an example.

Example 2.1. Let $\Pi_1 = \{4, 5, A, B\}$ and $\Pi_2 = \{8, F, G, Z\}$. Let $s_1 =$ `4445AABBA44` and $s_2 =$ `88FGZGGFZZ` be two strings respectively over Π_1

and Π_2. The values of π_1 and π_2 define the cardinality of each subset in a π-partition. For $\pi_1 = \pi_2 = 2$, one (of the many) possible matching schema is {{4,5}-{8,Z}, {A,B}-{F,G}}. Note that here {4,5}-{8,Z} means that symbols 4and 5match with symbols 8and Z. The best alignment $\langle \bar{s}_1, \bar{s}_2 \rangle$ obtained by this matching schema is

s_1 : 4445AABBA44 → 4445AABBA44

s_2 : 88FGZGGFZZ → -88FGZGGFZZ

** * *****

which denotes that s_2 can be obtained from s_1 by applying 3 edit operations, giving $\mathcal{L}_{\langle 2,2 \rangle}(s_1, s_2) = 3$. Here the positions in which the pair $\langle \bar{s}_1, \bar{s}_2 \rangle$ does not have a match is denoted by a blank space. □

In order to simplify the notation, we will denote by $\mathcal{L}(s_1, s_2)$ the MPED obtained between s_1 and s_2. Moreover, observe that values of $\mathcal{L}(s_1, s_2)$ may vary between 0 and the length of the longest string. In order to simplify the presentation, we will exploit a standardized version of the MPED, defined as follows:

$$\mathcal{L}^*(s_1, s_2) = \frac{\mathcal{L}(s_1, s_2)}{len(\bar{s}_1)}$$

which is defined over the interval [0.1]. Here $len(\bar{s}_1)$ (or equivalently $len(\bar{s}_2)$) is the length of the optimal alignment computed for $\mathcal{L}(s_1, s_2)$.

2.2. Edge-based method for short-term anomaly detection

We now describe our novel online anomaly detection method, building on the concepts introduced in Section 2.1.1, specifically RBM_{OCD}, dubbed *Anomaly detection with Generative replay* (AnGe).

The sensor measurements occur at specific time intervals. At any specific time t, new measurements are given by all sensors of a node and they are collected in a vector $\mathbf{x}^t$. Starting with $t = 0$ in a continuous loop, an RBM^t_{OCD} is trained online to model all measurements made until t. At the same time, we know [33–36] that the reconstruction error of unseen data points with an offline trained RBM gives a similarity metric with respect to the training data points. Thus, our assumption is that if at the specific time t an anomaly happens then the reconstruction error of the measurements $\mathbf{x}^t$ will be very different from the reconstruction error of $\mathbf{x}^{t-1}$, both being reconstructed with RBM^{t-1}_{OCD}. The larger this difference, the higher is the chance of an anomaly. To quantify, let us denote this metric m_{AnGe}. Using Root Mean Square Error (RMSE) for the reconstruction error, it can be computed as follows:

$$m_{AnGe} = \sqrt{\frac{1}{n_v}\sum_{i=1}^{n_v}(\widehat{\mathbf{x}^t_i} - \mathbf{x}^t_i)^2} - \sqrt{\frac{1}{n_v}\sum_{i=1}^{n_v}(\widehat{\mathbf{x}^{t-1}_i} - \mathbf{x}^{t-1}_i)^2} \quad (2)$$

Further on, if more similar measurements with $\mathbf{x}^t$ will occur, $\text{RBM}^{>t}_{OCD}$ will gradually enlarge its encoded data distribution to incorporate also these types of measurements, so as not to consider them an anomaly anymore. It is worth highlighting that AnGe needs to store in the device memory just the RBM_{OCD} weighted connections. This makes it a suited to perform online anomaly detection in wireless nodes.

2.3. Cloud-based method for long-term anomaly detection

In order to formally describe the cloud-based method proposed in this paper, we first introduce a formal representation of data streams in terms of chunks of sequences, generated at given time intervals. Then, we present the formalization of the approach, which is composed of a training phase and of an operating phase.

Let $\mathcal{N}$ be a set of nodes and $\mathcal{S}$ a set of sensors. Each sensor $s \in \mathcal{S}$ is equipped on a node $n \in \mathcal{N}$ which might accommodate several sensors. In order to simplify the notation, we assume that each sensor $s \in \mathcal{S}$ is uniquely identified in the set and, if necessary, function $\gamma : \mathcal{S} \to \mathcal{N}$ returns the node n the sensor s is equipped on.

A generic sensor s periodically collects data; we define an *observation* as the value v collected by a sensor s in a specific time instant t, and we denote it as $\alpha_s(t)$. We assume t stores the complete timestamp of the collection (date/time).

A certain set of sensors is run for an arbitrary amount of time T; an arbitrary sequence of time instances $t_i, t_{i+1}, \ldots, t_{i+k-1}$ defines an *interval* over which a chunk of data (an ordered sequence of observations) is collected; this must be transformed into a string in order to apply MPED. Moreover, in order to analyze the behaviour of sensors, it is important to organize observations in specific time intervals.

In the application context of the present paper, we analyze data by hours and days. In particular, assume that observations span over a set of days $d \in [1..D]$, and that each day d is subdivided in hours $h \in [1..24]$, given function $\rho(t_i)$ which provides the hour h the time instant t_i belongs to and the function $\delta(t_i)$ which provides the day t_i belongs to, the sequence of time instants belonging to a certain hour h of a certain day d is formalized by:

$$\Upsilon(d, h) = \{t_i \mid t_i \in T, \rho(t_i) = h \text{ and } \delta(t_i) = d\}$$

which forms the basis for the construction of strings to be provided to MPED, which is formalized next.

Given a sensor s_i, a day of interest d and an hour of interest h, the corresponding sequence of observations is the ordered sequence:

$$q(s_i, d, h) = \{\Psi(\alpha_{s_i}(t_k)) \mid t_k \in \Upsilon(d, h)\}$$

where function Ψ transforms each single observation in the corresponding symbolic representation.

Clearly, observations can be composed over several hours, or several days, if needed.

2.3.1. The workflow of the cloud-based method

The workflow of the cloud-based method is in two phases. The first one is devoted to training the system under "normal" operational conditions, in order to understand expected information for each sensor and for each time slot. The second phase, applies information learned in the first phase to identify potential anomalies.

Intuitively, one of the novelties of the approach for long-term anomaly detection introduced in this paper, relies on the fact that the training phase does not compute expected *values* for the various sensors, but expected *correlations* between sensors. In particular, in the training phase, for each sensor, and for each hour, we identify the so called *mate*-sensor, i.e. the most correlated sensor among the set for that time slot. This mate is used as a reference during the operating phase. In fact, whenever the correlation between the two significantly changes, a potential anomaly can be detected.

It is important to point out that MPED plays a crucial role in this approach, since the sensors that are being compared might be heterogeneous and correlations found between mate sensors might be completely unexpected (for example, light and temperature of sensors positioned in different points in space).

Next, we formalize the two phases of the approach.

Training phase. The training phase starts by computing the average correlation between each pair of sensors for each hour, within a fixed period of training days D^T. In particular, for each pair of sensors s_i, s_j and each hour $h \in [1.24]$, we define $C(s_i, s_j, h)$ as the average correlation over days in D^T. Formally:

$$\forall s_i, s_j, h \;\; C(s_i, s_j, h) = \underset{d \in D^T}{\text{avg}} \{1 - \mathcal{L}^*(q(s_i, d, h), q(s_j, d, h))\}.$$

Based on C, for each sensor s_i and each hour h, we can formally define the *mate sensor* $s^*_{i,h}$ of s_i as:

$$s^*_{i,h} = \tau(s_i, h) = \underset{s_j}{\text{argmax}} \{C(s_i, s_j, h)\}.$$

Table 1
Example of average correlation between values from eight sensors $S1$, $S2$, $S3$, $S4$, $S5$, $S6$, $S7$, and $S8$ during N days for time interval number 1. In boldface the highest correlation between each couple of sensors.

	$S1$	$S2$	$S3$	$S4$	$S5$	$S6$	$S7$	$S8$
$S1$	–	**0.9**	0.3	0.8	0.5	0.3	0.5	0.8
$S2$	**0.9**	–	0.5	0.9	0.5	0.4	0.8	0.2
$S3$	0.3	0.5	–	**0.9**	0.4	0.4	0.6	0.8
$S4$	0.8	0.8	**0.9**	–	0.7	0.5	0.7	0.1
$S5$	0.5	0.5	0.4	0.7	–	0.3	**0.8**	0.4
$S6$	0.3	0.4	0.4	0.5	0.3	–	0.3	**0.6**
$S7$	0.5	0.8	0.6	0.7	**0.8**	0.3	–	0.5
$S8$	0.8	0.2	0.8	0.1	0.4	**0.6**	0.5	–

Finally, for each sensor s_i and each hour h, we define the *expected correlation* of sensor s_i at hour h with its *mate* as $\eta(s_i, h) = C(s_i, \tau(s_i, h), h)$.

As an example, Table 1 shows the correlation computed during N days for eight heterogeneous sensors for $h = 1$.

Since in the one hour time interval the sensor $S1$ and the sensor $S2$ are the best correlating ones, it is reasonable to expect similar levels of correlation for corresponding time intervals in days different from the N considered ones. The mating between sensors that is extracted from the analysis of Table 1 for $h = 1$ is ($S1 \leftarrow\rightarrow S2$, $S3 \leftarrow\rightarrow S4$, $S5 \leftarrow\rightarrow S7$, $S6 \leftarrow\rightarrow S8$). A similar computation is carried out for the other values of h.

Operational phase. The operational phase starts after the training phase is completed. Here, each sensor has been already associated with its mate. Thus, the operational phase works as follows.

Given a threshold $\theta \in [0, 1]$, for each sensor s_i, for each day d, and for each hour h, we compute the *actual correlation*, denoted by $\chi(s_i, h, d)$, as the correlation between sensors s_i and its mate $\tau(s_i, h)$. Formally:

$$\forall s_i, d, h \quad \chi(s_i, h, d) = 1 - \mathcal{L}^*(q(s_i, d, h), q(\tau(s_i, h), d, h)).$$

Now, a potentially anomalous behavior is detected when the actual correlation of s_i with its mate, significantly differs from the expected one:

$$|\chi(s_i, h, d) - \eta(s_i, h)| > \theta.$$

In order to reduce false positives, an alert is issued if this condition is verified for an average difference greater than the threshold for a fixed number of hours H^*. Formally:

$$alert(s_i, h, d) \leftarrow \operatorname*{avg}_{h' \in [h-H^*, h]} \{|\chi(s_i, h', d) - \eta(s_i, h')|\} > \theta.$$

Observe that, choosing only one mate per sensor allows us to reduce the computational requirements of the approach. In fact, when it is needed to verify the behaviour of a sensor in a certain time slot, its data must be compared only with the data from another sensor, the mate, in the same slot. In Section 3 we show that this choice allows to provide also good performances in terms of capability in detecting anomalous behavior. A more complex approach could consider grouping each set of similar sensors in a "mate" cluster; in this way, checking the behavior of a sensor would result in comparing it with all the sensors in the mate cluster. On the one hand this solution could mitigate detection errors due to potential malfunctioning of the single mate sensor; however, in our opinion, this would significantly increase computational requirements of this phase.

2.4. Discussion on the improvements with respect to previous work and overhead of the proposed approach

Next, we discuss improvements of the present proposal with respect to previous work. We elaborate on the overhead of the proposed approach, considering both the edge and the cloud-based methods.

2.4.1. Discussion on the edge-based method

All earlier works which use machine learning to perform edge base anomaly detection, e.g. [20–23], need to store, besides the model parameters, the historical measurements on the device, in order to be able to adapt online and perform anomaly detection. This can lead to serious memory requirements which usually are not available on low-resource platforms. Our proposed method, i.e. AnGe, addresses this issue in effectively. Instead of storing the historical measurements in the device memory, it learns their distribution in an online manner within the RBM model, and then generates approximations of those measurements. To do so, AnGe makes use of our previous work, i.e. generative replay [24], to be able to learn data distributions in an online manner. Generative replay [24] is a general learning method for online learning which was not capable of performing anomaly detection. In this paper, we address this aspect, and we propose AnGe which is capable of performing online anomaly detection.

Thus, AnGe needs to store just the model parameters in the device memory. For instance, in the specific case of the experiments performed in this paper, AnGe stores 43 real-valued numbers on each node, which represent the connection weights between RBM neurons and their biases. Also, in terms of computational time, at each time step, AnGe has to perform about 330 multiplications, and about 330 summations of two real-valued numbers. As discussed and shown later in Section 3.2.1, these values add just a very small overhead to the battery lifetime of the sensors.

2.4.2. Discussion on the cloud-based method

In this section we point out differences and improvements of the cloud-based method proposed in this work with respect to the one presented in [18]. First of all, the previous approach was conceived to identify (possibly hidden) correlations between heterogeneous sensors. The introduction of MPED was the key factor for the success of the approach. As pointed out in [18] the computation of such a correlation, and the observation of a significant variation of it, may contribute in identifying an anomaly. However, anomaly detection has not been elaborated in [18], which is the specific topic addressed in the present work. Moreover, the approach proposed in [18] needs to compute MPED on the entire history of sensor data and needs to compare all pairs of sensors in the WSN. These issues have been resolved in the present work, by the definition of a mate sensor, which allows to compare data of only one pair of sensors, and by applying MPED to only one chunk of data at a time. As better explained next, these improvements significantly reduce the overhead of the new approach.

In order to evaluate the overhead of the cloud-based method, we next briefly compare the computational complexity of [18] and of the present approach. First of all, in both cases the computation of MPED is the most expensive task. In fact, even if MPED is computed by an heuristic approach [18], a quadratic dependency on string lengths is still required due to the computation of the edit distance. Intuitively, the complexity of this task can be expressed as $O(\iota \times len(s)^2)$, where ι is the number of iterations required by the heuristics, and can be tuned, whereas $len(s)$ is the maximum length of input strings. As previously pointed out, in [18] each MPED must be computed on the entire history of the sensors; consequently $len(s)$ may become quite large. On the contrary, in the present work each MPED is computed only on a portion of fixed length $\bar{l}$ corresponding to one hour of operation. In [18] MPED must be computed for all the pairs of sensors, whereas here only the MPED between the sensor under observation and its mate must be considered. In particular, in order to issue an alert, the complexity is, intuitively, $O(H^* \times \iota \times \bar{l}^2)$. As a consequence, the computational improvement of the current approach can be estimated by considering that $\bar{l}$ is much smaller than $len(s)$ and that H^* can be tuned and is usually a small number (we used $H^* = 6$ in our experiments).

It is worth observing that the various parameters of the approach can be tuned to obtain a proper balance between accuracy and execution time based on the available hardware and sensor network.

3. Experimental analysis

In this section we present some experiments we carried out in order to evaluate the effectiveness of the approach. We designed a test case with a heterogeneous WSN including sensors working in different areas of a building. We added different kind of synthetic interferences - the specific test case is detailed in the next subsections. The objectives of these tests are manifold: *(i)* check which kind of anomalies the edge-based method is able to identify in the given test case; *(ii)* check which kind of anomalies the cloud-based method is able to identify in the given test case; *(iii)* verify if the edge-based method would be enough to detect all interesting anomalies; *(iv)* verify if the conditional activation of the cloud-based method by alerts issued from the edge-based method would be enough to detect interesting anomalies.

In particular, it is interesting to check the correspondence between artificially inserted interferences and anomalies detected by both approaches; during this task, it is important to consider also the actual impact of the interference on the data sensed by the network. Similarly, it is important to consider possible anomalies detected outside the time-slots regarding the artificial ones and relate them to the testing environment.

In the following sections we first describe the test case and show collected data; then we analyze the results obtained by the edge-based method and by the cloud-based method separately. Next, we provide a discussion on the obtained results and on the advantages of combining the two. Finally, we present some possible validity threats of the approach.

3.1. Sensor network deployment setup

For the proposed experimentation, eight WSN nodes have been deployed in a floor at DIMES, cubo 41C, University of Calabria, Italy. The used WSN nodes consist of TelosB motes [37] running TinyOS 2.1.2 [38]. Such nodes have been organized in a multi-hop WSN by using the Building Management Framework (BMF) [39].

The BMF is a domain-specific framework specifically designed to efficiently manage heterogeneous WSNs that have been scattered in buildings. Through the BMF it is possible to quickly prototype WSN applications, realize smart sensing/actuation, and capture, by using specific abstractions, the floor plan of a building. BMF WSNs are controlled through a basestation that can be seen as both a data collector and a network configurator. BMF nodes communicate by using the BMF Communication Protocol, namely an application level protocol built on the Collection Tree and Dissemination Protocols [40,41].

In Fig. 2, an example of a BMF network together with the BMF layers at both basestation and node sides is portrayed.

The BMF has been here used to collect every second data from light, temperature and humidity sensors, compute on the nodes the average on such data, and to send the results every minute to a BMF basestation. The BMF basestation has been enhanced with a specific filter to clean redundant packets received from the WSN and to mask packet losses.

Fig. 3 shows all the nodes deployed and their location on the floor plan of the building involved. In particular, based on their location, the deployed nodes have been grouped in pairs:

- *nodes 1 and 124* are stuck on the window of an office. These nodes can be reached by direct sunlight;
- *nodes 17 and 27* are placed over a bookcase in an air conditioned office. Such nodes are less influenced, with respect to nodes 1 and 124, by direct sunlight;
- *nodes 25 and 31* are placed over a desk in an air conditioned and artificially illuminated laboratory.

The experimental tests that have been carried out and that span over 27 days, are divided in three parts, of 9 days each:

- In the first part all the nodes worked in a normal situation (no induced interferences) and are powered.
- In the second part, some interferences are introduced at the nodes 1, 17, 31, and 5. In particular, node 1 has been covered with a thick sheet of paper and a bag full of silicon has been placed close to it; a lighted bulb has been located adjacently to nodes 17 and 31; a bag full of silicon has been posed close to node 5.
- In the third part, no node has been subject to interferences. However, nodes 1, 17, 25, and 28 have been battery powered.

Raw data from sensors are shown in Figs. 4–6. In the deployment of the experiments, we first carried out the training phase of the approach for long-term anomaly detection defined in Section 2.3.1 by setting the fixed period of training days D^T to the first three days of nodes total acquisition time. In the fixed period, nodes worked in a normal situation with no external interferences and were powered. In this work, we set a threshold $\theta = 0.25$ for the operating phase and the parameter H^* defined in Section 2.3.1 has been set to 6 hours. It is worth pointing out that these values have been experimentally tuned for the analyzed test case and should not be considered as general values valid in every experimental setting. In particular, the training time D^T depends on the length of expected operativity of the system, and on the variability of sensed data. In fact, a long expected operativity and a high variability should suggest a longer training period than the one adopted in these tests. Similarly, values of θ and H^* must be chosen to tune the sensitivity to data variations; in fact, low values of these two parameters make the system more prone to issue alerts, whereas with high values of these parameters the system might miss some anomalies.

3.2. Experimental analysis of the edge-based method for short-term anomaly detection

In this subsection, we analyze the behavior of the online anomaly detection algorithm, i.e. AnGe, described in Section 2.2. We run the algorithm for each node separately, considering the measurements of all sensors for a node.

The RBM_{OCD} model was set to have 3 visible neurons and 10 hidden neurons. This yields a total of 43 parameters which have to be stored in the memory. The model parameters have been updated after each measurement in an online and continuous manner. Before each update, three samples were generated by the current model to avoid catastrophic forgetting during the learning of the new data measured.

Fig. 7 shows for each node separately how AnGe is capable to detect anomalies at each time step. For instance, let us consider the Subplot Fig. 7(c) which corresponds to the node 17. Usually, m_{AnGe} is very close to zero suggesting that there are no anomalies at that time step, while sometimes it is very far from zero suggesting strong anomalies, e.g. around the 8000 minute m_{AnGe} shows high oscillations and values ranging between -100 and +100. This is exactly the moment when the node 17 starts to be exposed to artificial interferences, i.e. a light bulb in its physical neighborhood. This is reflected by the new pattern of the sensor measurements. Further on, a bit before the 20,000 minm_{AnGe} shows strongly again the possible apparition of an anomaly, reaching a value of -500. This is exactly the moment when the light bulb was removed from the neighborhood of the node 17. Similarly, it can be clearly observed for the other nodes which have been exposed to artificial interferences, i.e. 1, 31, and 5, how AnGe detects the artificially introduced anomalies. Moreover, it is interesting to see how AnGe corresponding to the nodes which were not exposed directly to the artificial interferences, but which were close enough to the nodes with artificial interferences, can also detect them. A more spiky behavior of AnGe can be observed for nodes 1 and 124. These can be explained by the fact that they were exposed to several unknown interferences as their environment was not perfectly controlled (i.e. direct exposure to sunlight).

It is worth noting that in this paper we did not consider necessary to put thresholds on m_{AnGe} values as our goal was to show how AnGe is capable to detect big, but also small, changes in measurements patterns.

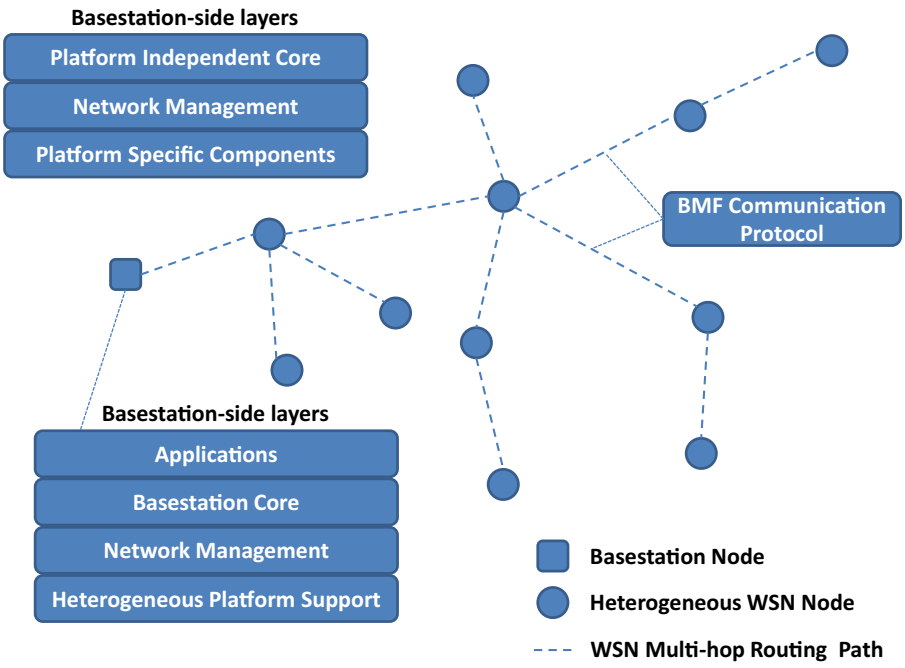


Fig. 2. A BMF network example.

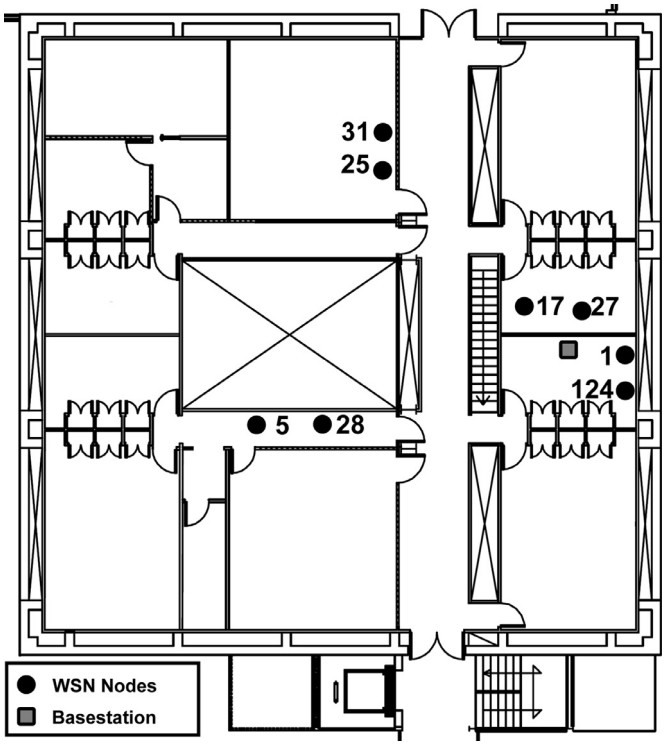


Fig. 3. Floor plan and nodes with corresponding identifiers for the experimental analysis of the proposed framework.

If one would use thresholds then it could easily control the sensitivity of AnGe based on the application requirements.

3.2.1. Analysis of energy consumption and battery duration

In this subsection, an estimation of the battery duration of TelosBs running AnGe (Section 2.2) will be given. First of all, it is worth noting that the radio is the main actor regarding the energy consumption of sensor nodes. In particular, it has been shown in the literature that the radio has a consumption ten times greater than CPU processes [42,43]. Given this, it is very important the use of a tool such as the BMF [39] providing data aggregation on the nodes and allowing the effective and efficient management of the radio duty cycle. Since AnGe has been implemented as a BMF Function, it just adds a small overhead to the BMF energy consumption. Fig. 8 shows the estimation of the mean lifetime at different duty cycles of a node which, respectively, (i) every second reads a sample from the desired sensors and sends a packet to its basestation, (ii) sends a packet every minute containing the average of the data gathered every second from such sensors, and (iii) adds the AnGe Function to the point number (ii) and, eventually, sends the anomaly detected. As can be seen, the BMF allows for a significant battery saving by sending synthetic data every minute; and the AnGe function does not add a notable load to the standard task of data collection. In the explained estimations, the TelosBs have been considered as powered by two 2700 mAh batteries and no supplemental energy consumptions (besides radio, CPU, and sensors) have been considered in the nodes.

3.3. Experimental analysis of the cloud-based method for long-term anomaly detection

Figs. 9–11 show the values of $|\chi(s_i, h', d) - \eta(s_i, h')|$, defined in Section 2.3.1, for light, temperature and humidity sensors, respectively. In order to simplify the presentation, only values greater than the threshold are shown; these would correspond to an alert. In this case, setting a threshold was important since the formula measures even smooth differences between expected and computed values and, consequently, without a threshold it would have been difficult to read the graphs. Obviously, also in this case, an accurate tuning of the threshold could easily control the sensitivity of the approach.

Let us first consider light values. In particular, as far as node 1, it is interesting to observe that, even if an artificial interference was added, no long-term anomaly is alerted. This result is actually correct. In fact, the thick sheet of paper added in front of the sensor reduces the amount of light perceived by the sensor, but it does not prevent it to detect external light variations over long terms. This is also consistent with the result obtained by the short-term approach, which identifies many small environment interferences. On the contrary, nodes 17 and 31, which were disturbed by a lighted bulb, became almost unable to detect light

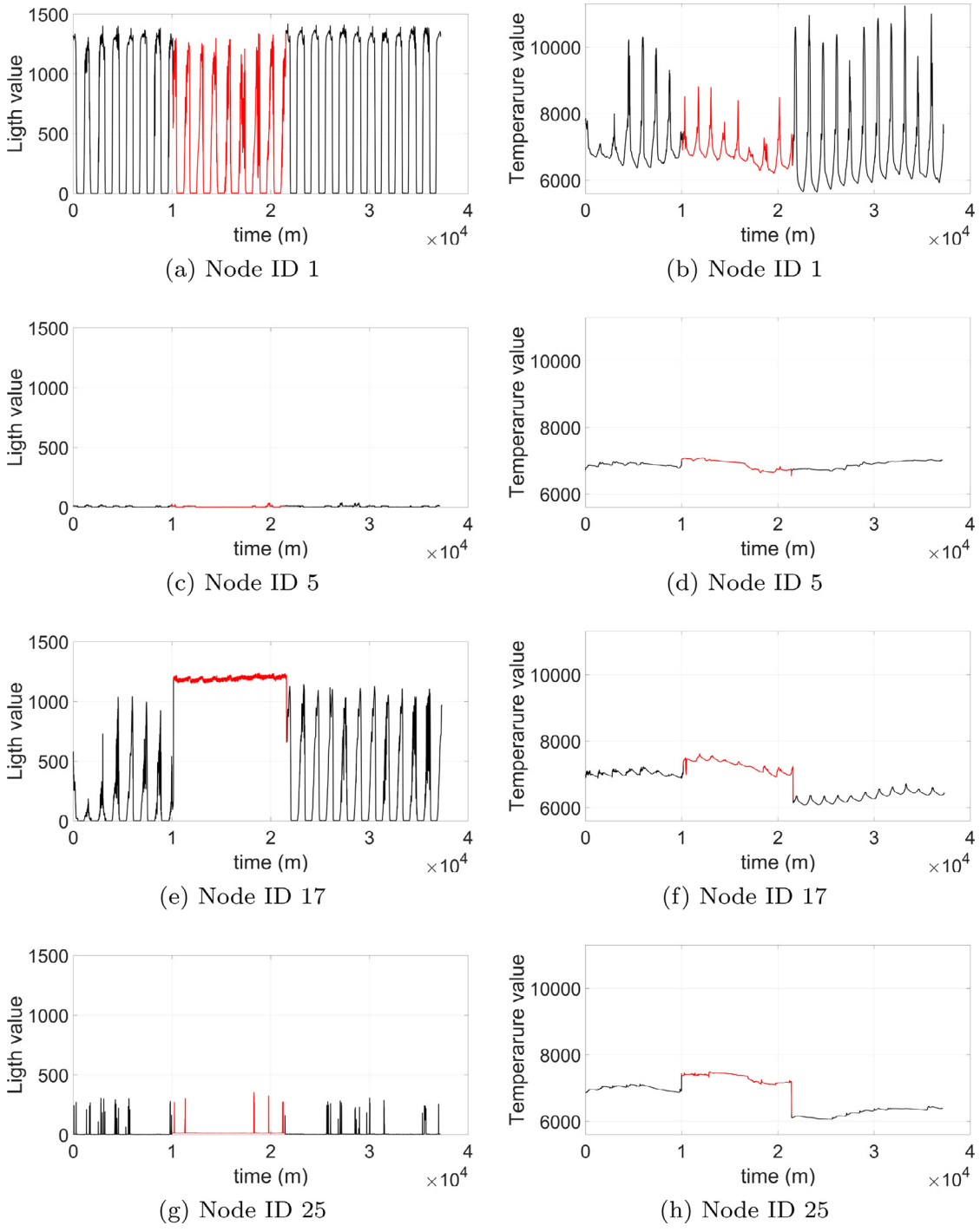


Fig. 4. Light and temperature raw sensor data for node identifiers 1, 5, 17, and 25. The temporal window corresponding to the application of interferences to sensors 1, 5, 17, and 31 is highlighted in red for all the plots. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

variations (see Figs. 4 and 5); and in fact a long-term alert during all the test period is fired.

It is interesting to stress that, in a possible application of the short-long term combined approach, the short-term method activates the long-term one, which may confirm the persistence of an anomalous situation or it may categorize the alert just as occasional.

Interestingly, node 124 is completely unaffected by long-term anomalies; this is also right because it did not receive external interferences and the interference on the adjacent sensor is totally local (a sheet of paper). The same does not hold for nodes 25 and 27 which are near to sensors disturbed by a lighted bulb (17 and 31); as a consequence, they are slightly affected too. This result is again consistent with the results of the short-term approach.

Finally, as far as light results for nodes 5 and 28 are concerned, we can observe some spikes over all the period, but not particularly constant to motivate a long-term anomaly, especially during the artificial interference. This can be motivated by both the fact that they are positioned in a corridor which generates highly irregular data and by observing that, in this case, interference is about humidity caused by the bag full of silicon.

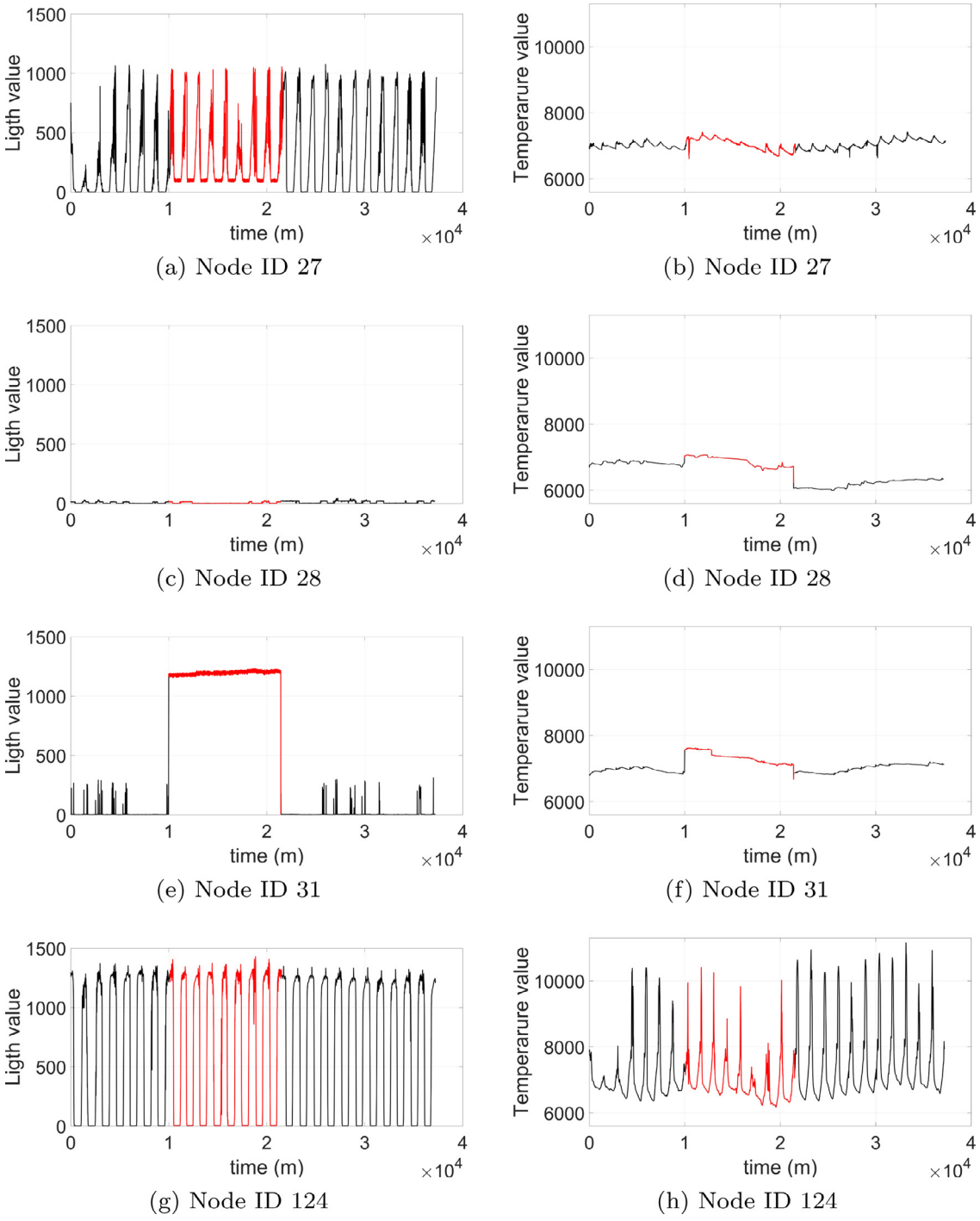


Fig. 5. Light and temperature raw sensor data for node identifiers 27, 28, 31, and 124. The temporal window corresponding to the application of interferences to sensors 1, 5, 17, and 31 is highlighted in red for all the plots. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Results for temperature show no particular alerts, except for node 25. This is again consistent, since no artificial interference on temperature was actually introduced, and the alerts on node 25 correspond to the last part of the experiment, when the node was battery powered.

Finally, as far as humidity is concerned, we observe consistent long-term alerts only on node 1, where a silicon bag was placed close to it. As for node 5, disturbed by the other silicon bag, we observe no long-term alerts. If we observe the raw data for humidity shown in Fig. 6 we may actually observe no particular variations in trend values. Again, this result is consistent also with short-term analysis, which was able to point out the time instants when the bag was put/removed beside the sensor.

3.3.1. *Analysis of performances and overhead of the cloud-based method*

Since the algorithm for computing MPED in the cloud-based method is the most computationally demanding task of the proposed approach, we next present an analysis of its performances and overhead. The objective of these tests is to show effectiveness and scalability of the proposed approach. All tests presented in this section have been executed on a server equipped with an Intel Xeon X3430 processor and 4 GB of

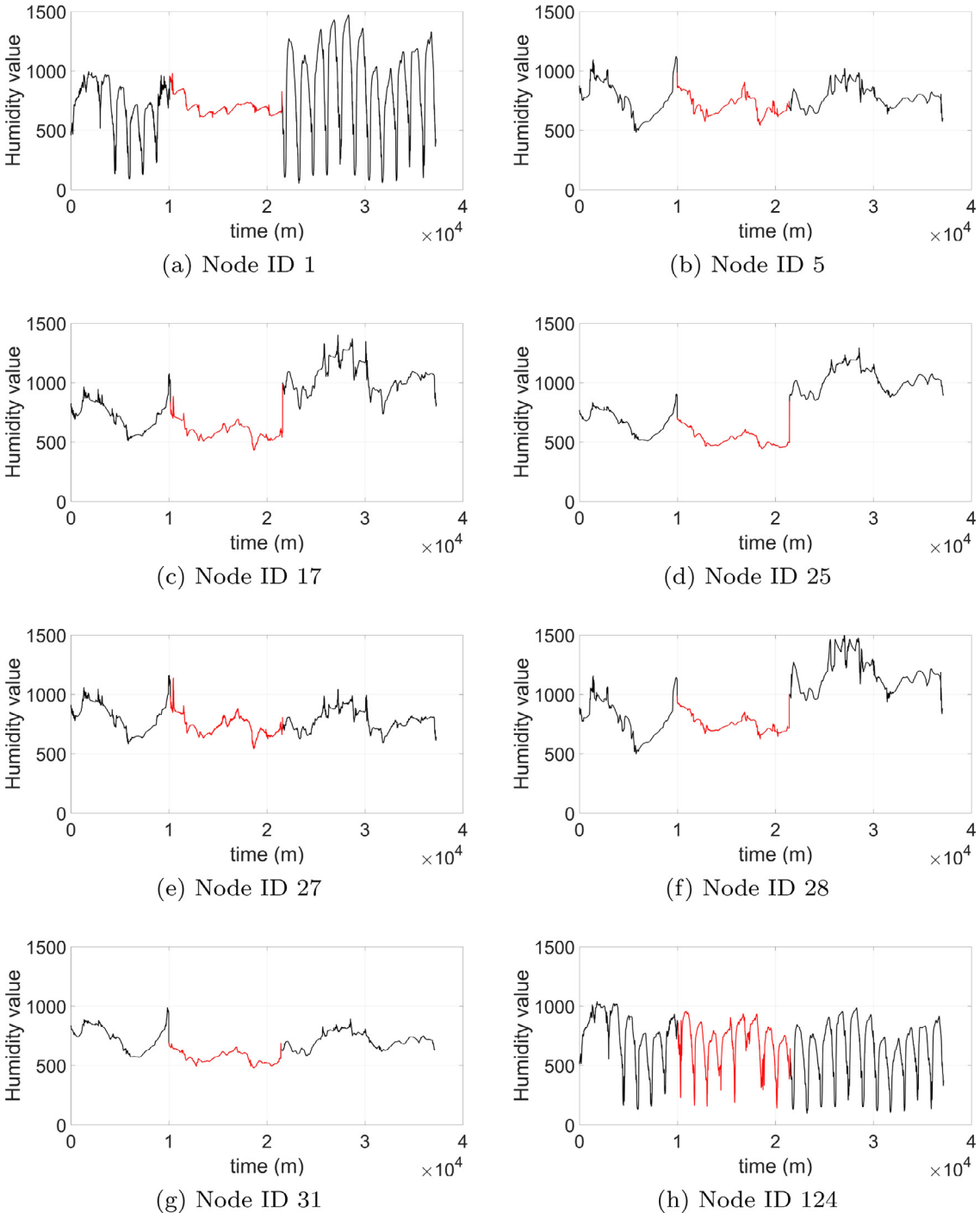
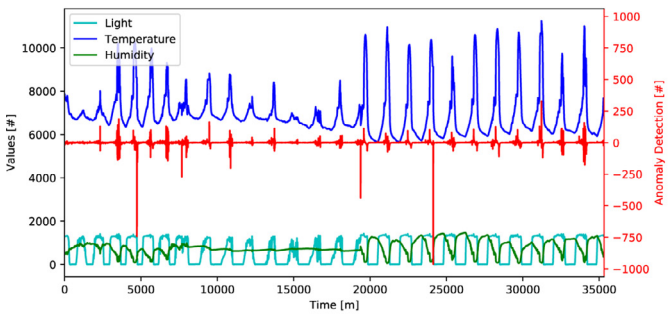


Fig. 6. Humidity raw sensor data for node identifiers 1, 5, 17, 25, 27, 28, 31, and 124. The temporal window corresponding to the application of interferences to sensors 1, 5, 17, and 31 is highlighted in red for all the plots. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

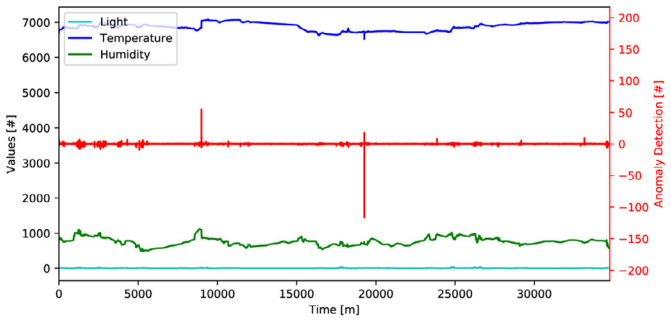
RAM running the Ubuntu Linux kernel 2.6.26-2-686-bigmem SMP i686 GNU/Linux operating system

First of all, it is worth recalling that the implementation of MPED computation needs to resort to some heuristics, given the NP-Hardness of the problem. In the implementation of the method we considered several heuristics, based on different strategies. Here we show some results for the local search heuristic Steepest Ascent Hill Climbing algorithm with random restart (hereafter HC), and for the population-based metaheuristics Evolution Strategy (hereafter, ES). Performance comparison of these two heuristics for a given a complex set of input parameters is shown in Fig. 12, where it is clear that ES outperforms HC both for number of iterations needed to reach the best MPED value and for precision (it reaches a lower value of MPED than HC). A similar behavior can be observed for different input parameter sets.

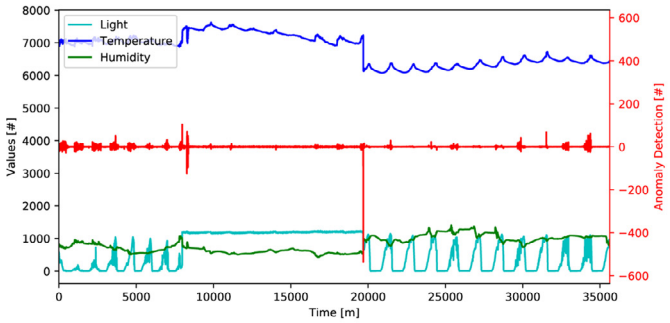
In order to verify if the good performances of ES actually do not degradate the quality of the computation, we evaluated its precision with respect to an exhaustive approach computing the exact solution. Precision is computed as follows: let d_{EX} be the MPED of an instance using the exhaustive approach, i.e., the optimum, and let d_{ES} be the



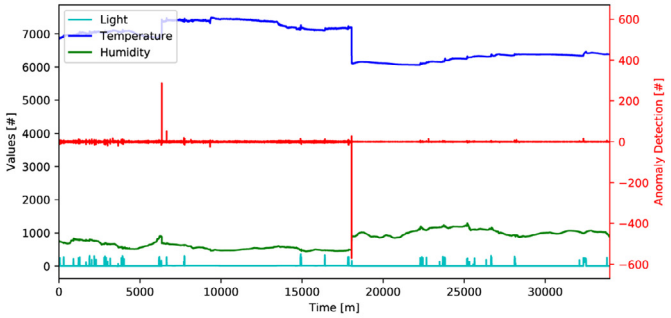
(a) Node ID 1



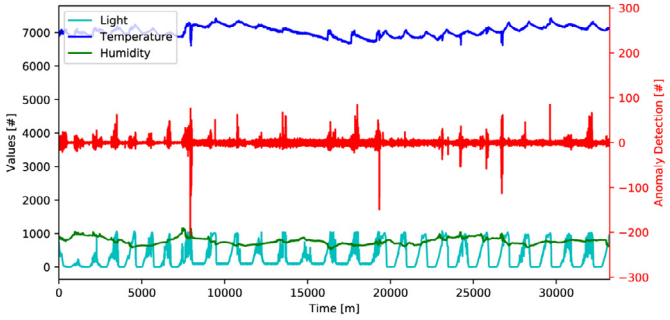
(b) Node ID 5



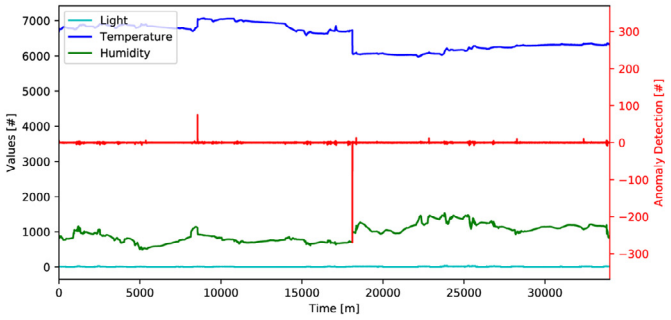
(c) Node ID 17



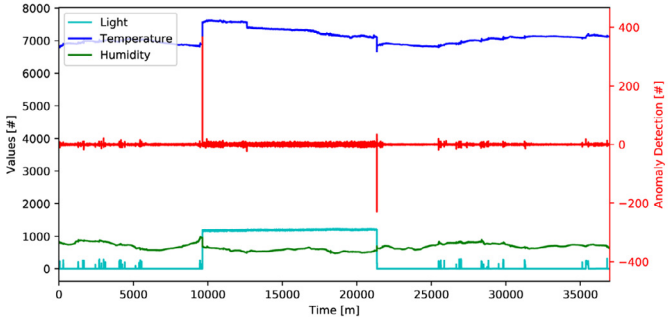
(d) Node ID 25



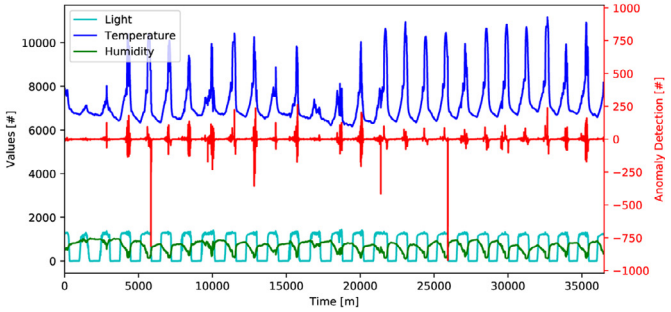
(e) Node ID 27



(f) Node ID 28



(g) Node ID 31



(h) Node ID 124

Fig. 7. Short-term anomaly detection. Each subplot reflects how our proposed method, Anomaly detection with Generative Replay (AnGe), detects anomalies on a specific node. The x-axes represent the time, the left y-axes show the sensors measurements, while the right y-axes (red) show the values of m_{AnGe}. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

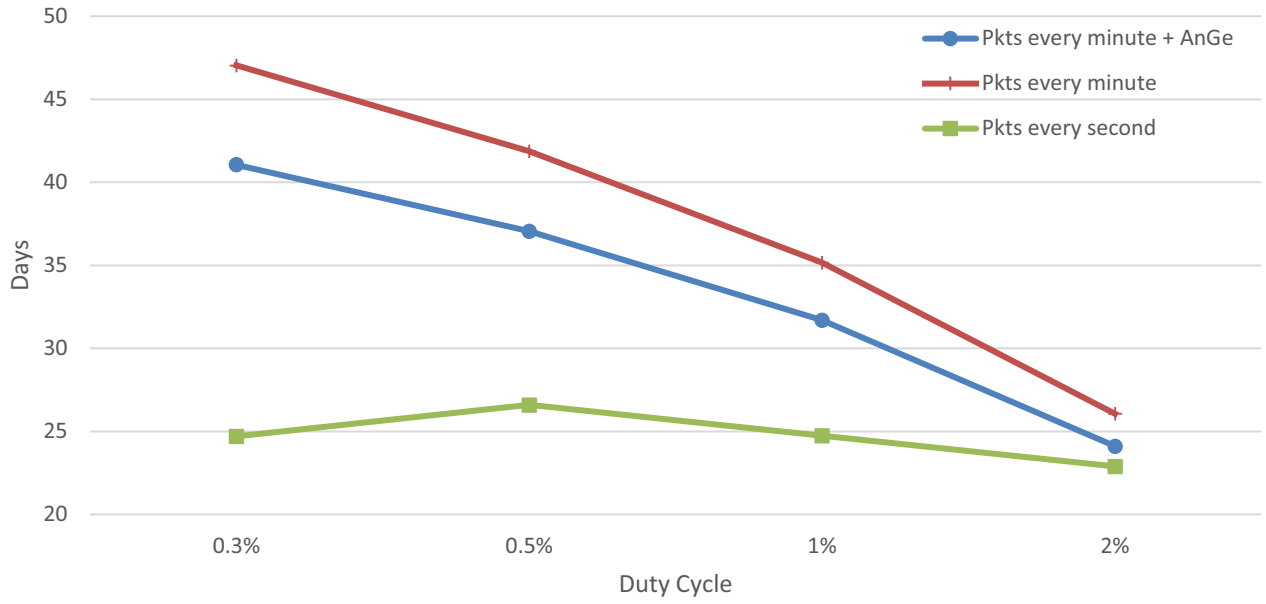


Fig. 8. Comparison of battery lifetimes, considering different duty cycles, for TelosBs running the BMF and (i) sending pkts per second, (ii) sending pkts per minute, and (iii) sending pkts per minute + elaborating the AnGe Function.

Table 2
Obtained precision P_{ES} of ES Heuristics.

		len(*s*)				
π	$\lvert\Pi\rvert$	500	1000	2000	3500	5000
	16	0.99	1.00	1.00	1.00	1.00
3	18	1.00	0.99	1.00	1.00	1.00
	20	1.00	1.00	1.00	1.00	1.00
	16	1.00	1.00	1.00	1.00	1.00
4	18	0.99	1.00	1.00	1.00	1.00
	20	0.99	0.99	1.00	1.00	1.00
	16	1.00	1.00	1.00	1.00	1.00
5	18	1.00	1.00	1.00	1.00	1.00
	20	1.00	1.00	1.00	1.00	1.00
	16	1.00	1.00	1.00	1.00	1.00
6	18	0.99	1.00	1.00	1.00	1.00
	20	1.00	0.99	1.00	1.00	1.00

MPED computed by the ES solution. We define precision P_ES as

$$P_{ES} = 1 - \frac{d_{ES} - d_{EX}}{d_{EX}} \tag{3}$$

Table 2 reports obtained results. Note that a value $P_{ES} = 1.00$ in the table indicates that ES reached precisely the same solution as the exhaustive approach. From the analysis of Table 2 it is possible to observe that ES always reaches a precision equal or very close to 1.00 on very different sets of input parameters. These results show the effectiveness of the approach in computing reliable MPED distances.

Finally, in order to evaluate the scalability of the approach, we measured the runtime of MPED computation for increasing string lengths and different values of alphabet sizes. Results are shown in Fig. 13 which demonstrates a good scalability of the approach and fairly acceptable execution times, below one second, even for the hardest configurations.

3.4. Discussion

The short-term approach clearly identifies potential anomalies signaled by the maxima values of $|m_{AnGe}|$ (Fig. 7(a)) for node 1, the node exposed to a thick sheet of paper. The long-term approach for node 1 signals an anomaly for humidity only (Fig. 11(a)). As explained in the previous section, this is consistent.

Node 124, the node close to node 1 but not exposed to any impairments, experiences several short-term alerts (Fig. 7(h)) but no long-term ones (Figs. 9(h), 10(h) and 11(h)). As a matter of facts, by analyzing the patterns of short-term alerts of nodes 1 and 124 we observe that they are almost identical, similarly to the overall trends of temperature, humidity and light measured by both sensors. It can be then concluded that short-term alerts were issued by the environment.

As far as node 5 is concerned, which was exposed to a bag full of silicon, the short-term approach issues two spikes for $|m_{AnGe}|$ (Fig. 7(b)), but no consistent long-term anomaly is issued (Figs. 9(b), 10(b) and 11(b)). As a matter of facts, the short-term approach identifies the moments when the bag was placed and removed, but this did not alter the measurements for humidity, as shown in Fig. 6.

Node 5 is close to node 28, which experiences similar behavior on short-term analysis (cfr Fig. 7(d) and (f)). However, only small alerts on light for long-term anomalies are issued for this node (Fig. 9(f)), and these are mostly outside the artificial interference period.

For node 17, disturbed with a lighted bulb, the short-term approach properly identifies the beginning and the end of the interference period (see Fig. 7(c)) and the long-term approach confirms the anomaly for light sensor (see Fig. 9(c)) while issuing no alerts for temperature and humidity (Figs. 10(c) and 11(c)). A similar behavior is observed on node 27 (Figs. 7(e), 9(e), 10(e) and 11(e)) which was close to node 17 and, consequently, also influenced by the light of the bulb.

Also for node 31, the other node influenced by a lighted bulb, the approach properly identifies the interference, with start and end points identified by the short-term approach (Fig. 7(g)) and interference period identified on light sensor by the long-term approach (Fig. 9(g)). In this case, for node 25, the one close to node 31, the short-term approach issues alerts (Fig. 7(d)) which are not confirmed by the long-term approach (Figs. 9(d), 10(d) and 11(d)), probably because the area where the nodes were positioned was much bigger than the area where nodes 17 and 27 were placed (see Fig. 3) and consequently, node 25 was less influenced by the nearby light on node 31.

Almost no alerts are issued in the period when the nodes were battery powered. As a matter of facts, looking at raw data shown in Figs. 4–6, no real variations in sensed data can be observed in this case.

Summarizing the overall results, we can observe that experiments confirm the intuition about the different nature of anomalies detected by the two approaches. These can be seen as complementary tools for

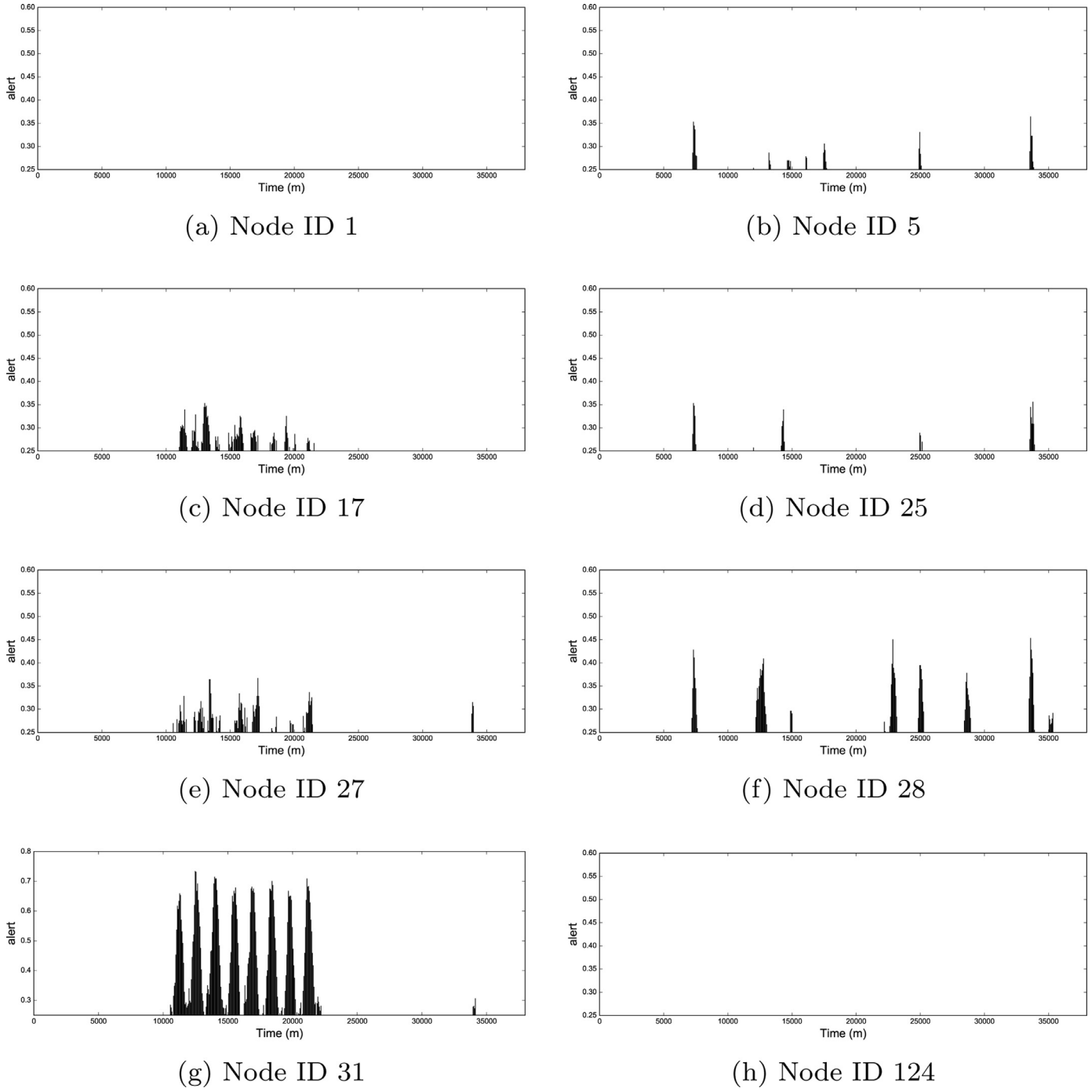


Fig. 9. Alerts (with over-threshold values) for long-term anomaly detection of light sensors. The x-axis represents the time, the y-axis indicates the value of $|\chi(s_i, h', d) - \eta(s_i, h')|$.

anomaly detection. Both correctly detect real anomalies at different stages. However, both are affected by false positive anomaly detection phenomena. This problem can be significantly reduced by using the short-term approach to "trigger" long-term observations, which can also drill down the analysis from nodes to single sensors.

3.5. Threat to validity

In the previous sections we showed that both the short-term method and the long-term method are able to detect anomalies that have been artificially inserted in the test case. Moreover, the combination of the two approaches allows to trigger the more computationally demanding task of long-term anomaly detection only when needed, and only on the sensors that need attention. Nevertheless, the long-term approach provides its results almost in real time, since it works on previously stored data and, consequently, the identification of relevant anomalies can be carried out promptly. Clearly, the approach is not intended to be a solution to every problem in anomaly detection; here we analyze potential weaknesses and limitations that may result in inability to identify anomalies.

First of all, it is worth recalling that the short-term method is based on an ANN and, as it usually happens in this context, its ability to properly classify anomalies strongly depends on the kind of signals it is trained on. Then, it may happen that the short-term approach misses

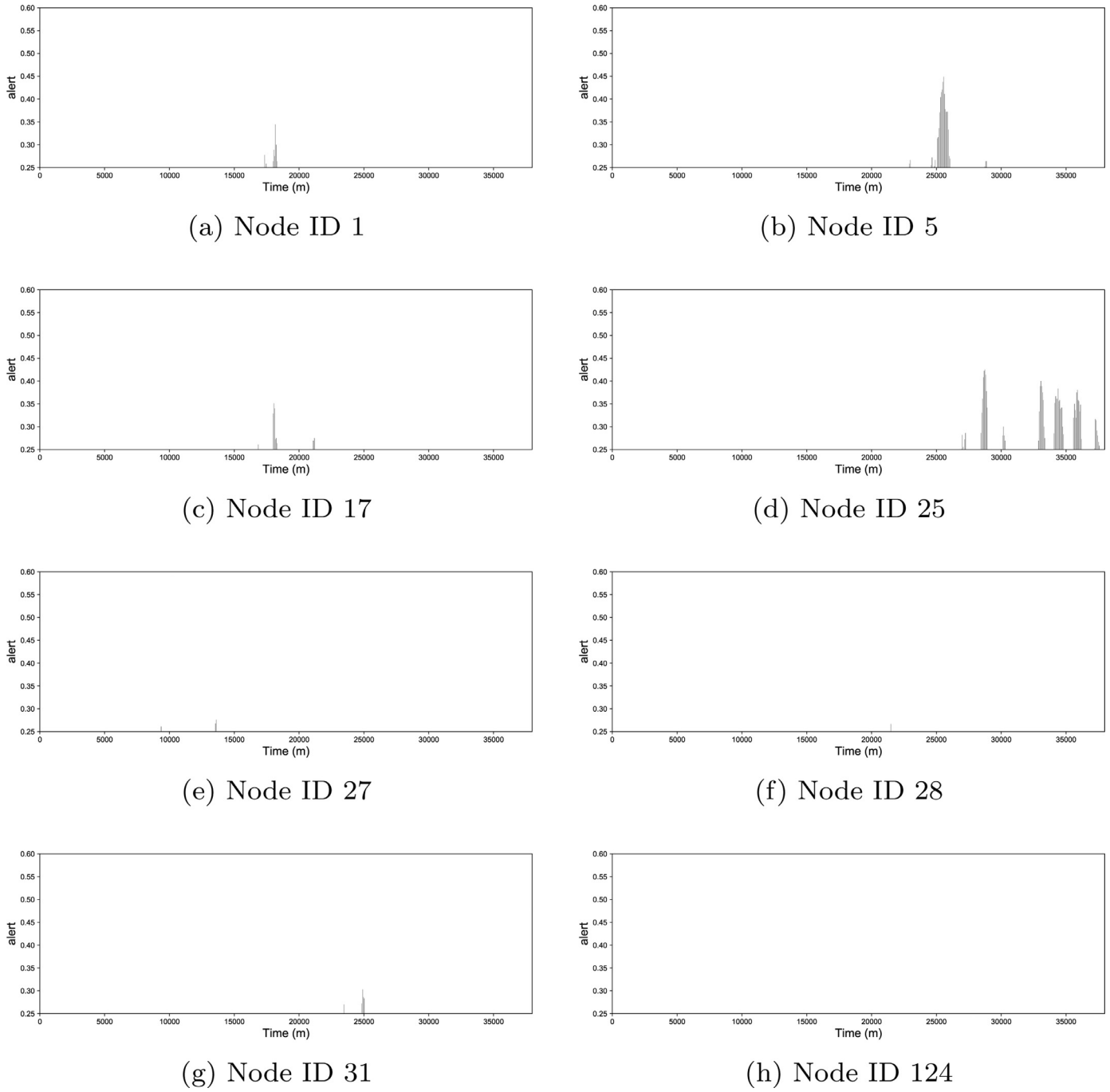


Fig. 10. Alerts (with over-threshold values) for long-term anomaly detection of temperature sensors. The x-axis represents the time, the y-axis indicates the value of $|\chi(s_i, h', d) - \eta(s_i, h')|$.

some kind of anomalies, such as drifts. Moreover, due to the intrinsic nature of the method, it may be in trouble in identifying "slowly appearing" anomalies, i.e. anomalies that start with very smooth changes in the data. Observe that, in our approach, if the short-term method does not identify an anomaly, then the long-term method is not activated. As a consequence, the ability to identify certain kinds of anomalies may be limited by this fact.

On the long-term method side, it is worth pointing out that its ability to identify anomalies is significantly related to the presence of good correlations between pairs of sensors, even if such correlations may be not obvious or evident. As explained in previous sections, related sensors must not necessarily be near in space or sensing the same measure; moreover, each sensor may have a different mate in different time slots. As a consequence, the possibility to identify a strong correlation between each pair of sensors is high, and it increases in complex networks including several nodes. However, there is still the possibility that correlations are weak for some sensors; in this case, the ability to identify real long-term anomalies may be flawed. In more detail, in case of a weak correlation, the difference between the expected data and the sensed one may easily be higher than a threshold, resulting in several

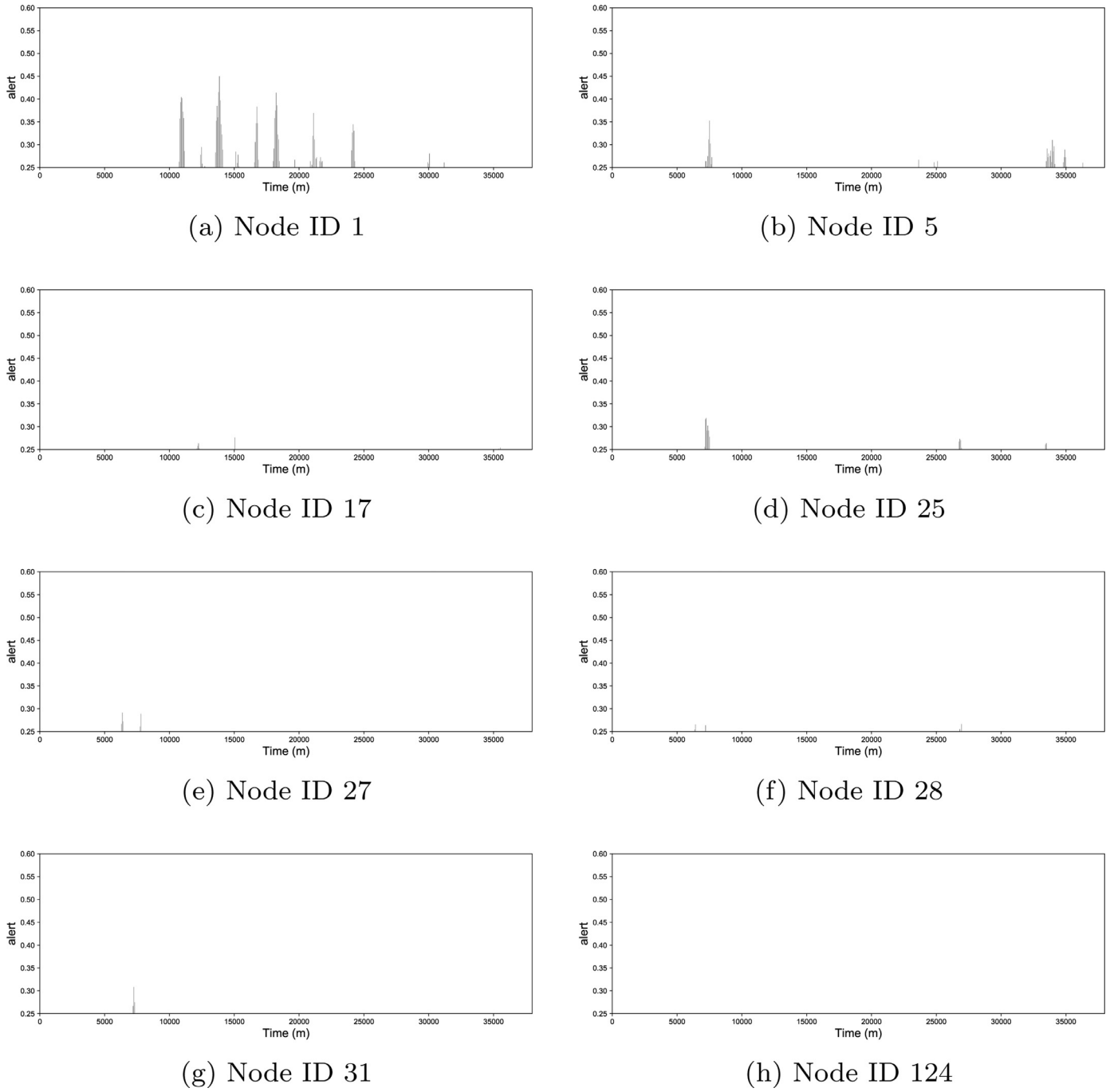


Fig. 11. Alerts (with over-threshold values) for long-term anomaly detection of humidity sensors. The x-axis represents the time, the y-axis indicates the value of $|\chi(s_i, h', d) - \eta(s_i, h')|$.

false positive long-term alerts. However, this problem is mitigated in our framework by the fact that the long-term method is activated only if a short-term anomaly is detected. Possible causes of a weak correlation are significantly variable signals missing any form of characterization.

4. Conclusions

Automatic anomaly detection in heterogeneous wireless sensor networks is a very challenging task. The signals captured by the sensors are affected by natural environmental variations that can mask signals variations caused by anomalies. A huge amount of information is captured by WSNs and there is a need to optimize the data analysis problem devising algorithms for local data preprocessing aiming at reducing the amount of data to be transferred for further processing.

The approach proposed in this paper explores how a combination of short-long term algorithms can correctly identify anomalies in a WSN. The proposed short-term approach has shown good performances for a local identification of potential anomalies and identifying temporal windows of potential interest to be transferred to a cloud service for further long-term analysis. The long-term approach has shown good performances for identifying the temporal windows affected by anomalies. Nodes close to each other can results in signalling a double alert nevertheless the impact of such false positive result is not so relevant considering that a manual intervention will affect a single location.

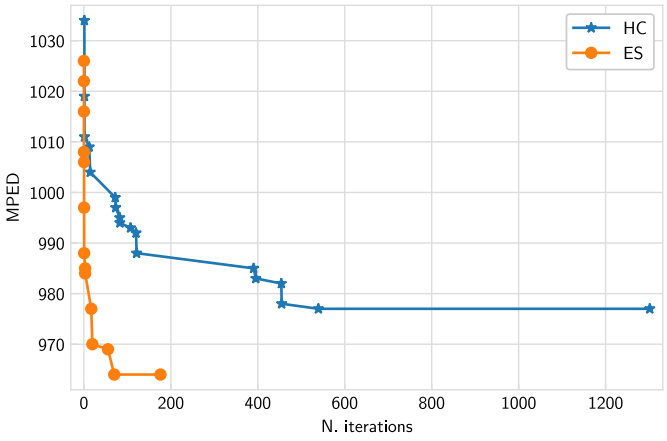


Fig. 12. Comparison of two heuristics for MPED, ES and HC, with $|\Pi| = 14$ $len(s) = 2000$ and $\pi_1 = \pi_2 = 4$.

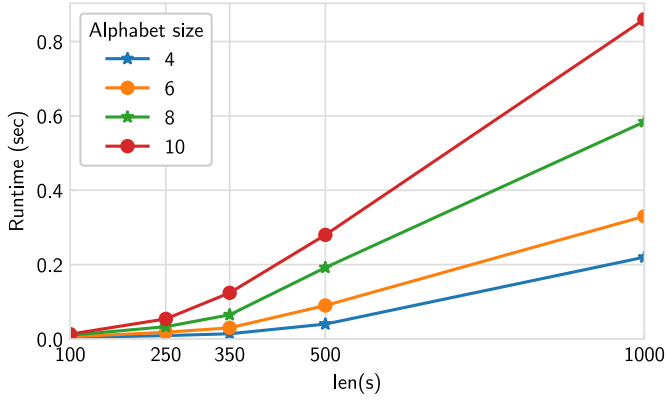


Fig. 13. Runtime of the computation of $\mathcal{L}_{(1,1)}(s_i, s_j)$ against $len(s)$ for different values of alphabet size $|\Pi|$.

Overall we demonstrated how a combined use of short-long term approaches may reduce the drawbacks of both, i.e. false positives and computational requirements, while taking advantage of the best qualities of both, i.e. timeliness and accuracy.

As far as future work is concerned, we plan to improve the coupling of the two approaches by automating the process and fine-tuning the thresholds. Moreover, the approach can also be improved to detect other kinds of anomalies that currently cannot be detected, like slowly changing values of sensed data; these are also known as drifts. To this purpose we are exploring the possibility to adapt a previous learning approach presented in [5], where it is shown that the ability to detect drifts by online approaches is strongly related to the learning rate and that usually the detection is made at the start of the drift, while the rest of the drift period is not flagged as anomalous. Our combined approach could overcome this issue. In fact, the online part, with the modified algorithms, may identify the start of the drift and then activate the offline approach which could flag the drift period.

Acknowledgements

This work was partially supported by: *(i)* the Italian Ministry for Economic Development (MISE) under the project "Smarter Solutions in the Big Data World", funded within the call "HORIZON2020" PON I&C 2014–2020; *(ii)* the "INTERIoT", Research and Innovation action - Horizon 2020 European Project, Grant Agreement #687283, financed by the European Union; *(iii)* "Smart-Hybrid: Piattaforma di gestione ottimale dei workload in ambiente cloud Ibrido", funded by POR Calabria FESR-FSE (CUP: J28C17000120006).

References

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, Wireless sensor networks: a survey, Comput. Netw. 38 (4) (2002) 393–422.
[2] G. Fortino, R. Giannantonio, R. Gravina, P. Kuryloski, R. Jafari, Enabling effective programming and flexible management of efficient body sensor network applications, IEEE Trans. Hum. Mach. Syst. 43 (1) (2013) 115–133, doi:10.1109/TSMCC.2012.2215852.
[3] G. Fortino, A. Guerrieri, G.M. O'Hare, A. Ruzzelli, A flexible building management framework based on wireless sensor and actuator networks, J. Netw. Comput. Appl. 35 (6) (2012) 1934–1952.
[4] J.A. Stankovic, When sensor and actuator networks cover the world, ETRI J. 30 (5) (2008) 627–633.
[5] H.H. Bosman, A. Liotta, G. Iacca, H. Wortche, Anomaly detection in sensor systems using lightweight machine learning, in: Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on, IEEE, 2013, pp. 7–13.
[6] Y. Zhang, J. Jiang, Bibliographical review on reconfigurable fault-tolerant control systems, Annu. Rev. Control 32 (2) (2008) 229–252.
[7] S. Ahmad, A. Lavin, S. Purdy, Z. Agha, Unsupervised real-time anomaly detection for streaming data, Neurocomputing 262 (2017) 134–147.
[8] Y. Yao, A. Sharma, L. Golubchik, R. Govindan, Online anomaly detection for sensor systems: a simple and efficient approach, Perform. Evaluat. 67 (11) (2010) 1059–1075.
[9] M. Xie, S. Han, B. Tian, S. Parvin, Anomaly detection in wireless sensor networks: a survey, J. Netw. Comput. Appl. 34 (4) (2011) 1302–1325.
[10] L. Akoglu, H. Tong, D. Koutra, Graph based anomaly detection and description: a survey, Data Min. Knowl. Discov. 29 (3) (2015) 626–688, doi:10.1007/s10618-014-0365-y.
[11] D. Savage, X. Zhang, X. Yu, P. Chou, Q. Wang, Anomaly detection in online social networks, Soc. Netw. 39 (2014) 62–70, doi:10.1016/j.socnet.2014.05.002.
[12] P. Garca-Teodoro, J. Daz-Verdejo, G. Maci-Fernndez, E. Vzquez, Anomaly-based network intrusion detection: techniques, systems and challenges, Comput. Secur. 28 (1) (2009) 18–28, doi:10.1016/j.cose.2008.08.003.
[13] C. Phua, V. Lee, K. Smith, R. Gayler, A comprehensive survey of data mining-based fraud detection research, arXiv preprintarXiv:1009.6119(2010).
[14] T. Ahmed, M. Coates, A. Lakhina, Multivariate online anomaly detection using kernel recursive least squares, in: IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications, 2007, pp. 625–633, doi:10.1109/INFCOM.2007.79.
[15] H.H.W.J. Bosman, G. Iacca, A. Tejada, H.J. Wörtche, A. Liotta, Spatial anomaly detection in sensor networks using neighborhood information, Inf. Fusion 33 (2017) 41–56.
[16] F. Serdio, E. Lughofer, K. Pichler, T. Buchegger, M. Pichler, H. Efendic, Fault detection in multi-sensor networks based on multivariate time-series models and orthogonal transformations, Inf. Fusion 20 (2014) 272–291.
[17] J. Greensmith, W. Aickelin, G. Tedesco, Information fusion for anomaly detection with the dendritic cell algorithm, Inf. Fusion 11 (1) (2010) 21–34.
[18] F. Cauteruccio, G. Fortino, A. Guerrieri, G. Terracina, Discovery of hidden correlations between heterogeneous wireless sensor data streams, in: International Conference on Internet and Distributed Computing Systems, Springer, 2014, pp. 383–395.
[19] H.H. Bosman, G. Iacca, A. Tejada, H.J. Wörtche, A. Liotta, Spatial anomaly detection in sensor networks using neighborhood information, Inf. Fusion 33 (2017) 41–56, doi:10.1016/j.inffus.2016.04.007.
[20] Y. Zhang, N. Meratnia, P.J. Havinga, Distributed online outlier detection in wireless sensor networks using ellipsoidal support vector machine, Ad Hoc Netw. 11 (3) (2013) 1062–1074, doi:10.1016/j.adhoc.2012.11.001.
[21] H.H.W.J. Bosman, A. Liotta, G. Iacca, H.J. Wrtche, Anomaly detection in sensor systems using lightweight machine learning, in: 2013 IEEE International Conference on Systems, Man, and Cybernetics, 2013, pp. 7–13, doi:10.1109/SMC.2013.9.
[22] H.H.W.J. Bosman, A. Liotta, G. Iacca, H.J. Wrtche, Online extreme learning on fixed-point sensor networks, in: 2013 IEEE 13th International Conference on Data Mining Workshops, 2013, pp. 319–326, doi:10.1109/ICDMW.2013.74.
[23] H.H.W.J. Bosman, G. Iacca, H.J. Wrtche, A. Liotta, Online fusion of incremental learning for wireless sensor networks, in: 2014 IEEE International Conference on Data Mining Workshop, 2014, pp. 525–532, doi:10.1109/ICDMW.2014.79.
[24] D.C. Mocanu, M.T. Vega, E. Eaton, P. Stone, A. Liotta, Online contrastive divergence with generative replay: experience replay without storing data, CoRR abs/1610.05555 (2016).
[25] H. Shin, J.K. Lee, J. Kim, J. Kim, Continual learning with deep generative replay, in: I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems 30, Curran Associates, Inc., 2017, pp. 2994–3003.
[26] N. Kamra, U. Gupta, Y. Liu, Deep generative dual memory network for continual learning, CoRR abs/1710.10368 (2017).
[27] P. Smolensky, Information processing in dynamical systems: foundations of harmony theory, in: D.E. Rumelhart, J.L. McClelland, et al. (Eds.), Parallel Distributed Processing: Volume 1: Foundations, MIT Press, Cambridge, 1987, pp. 194–281.
[28] H. Ackley, E. Hinton, J. Sejnowski, A learning algorithm for Boltzmann machines, Cogn. Sci. (1985) 147–169.
[29] G.W. Taylor, G.E. Hinton, S.T. Roweis, Two distributed-state models for generating high-dimensional time series, J. Mach. Learn. Res. 12 (2011) 1025–1068.
[30] J.L. McClelland, B.L. McNaughton, R.C. O'Reilly, Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory, Psychol. Rev. 102 (1995) 419–457.

[31] Y. Bengio, Learning deep architectures for ai, Found. Trends Mach. Learn. 2 (1) (2009) 1–127, doi:10.1561/2200000006.
[32] D.C. Mocanu, E. Mocanu, P.H. Nguyen, M. Gibescu, A. Liotta, A topological insight into restricted boltzmann machines, Mach. Learn. 104 (2) (2016) 243–270, doi:10.1007/s10994-016-5570-z.
[33] D. Mocanu, G. Exarchakos, H. Ammar, A. Liotta, Reduced reference image quality assessment via boltzmann machines, in: Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on, 2015, pp. 1278–1281, doi:10.1109/INM.2015.7140481.
[34] M.T. Vega, D.C. Mocanu, J. Famaey, S. Stavrou, A. Liotta, Deep learning for quality assessment in live video streaming, IEEE Signal Process. Lett. 24 (6) (2017) 736–740, doi:10.1109/LSP.2017.2691160.
[35] D. Mocanu, G. Exarchakos, A. Liotta, Deep learning for objective quality assessment of 3d images, in: Image Processing (ICIP), 2014 IEEE International Conference on, 2014, pp. 758–762, doi:10.1109/ICIP.2014.7025152.
[36] H.B. Ammar, E. Eaton, M.E. Taylor, D.C. Mocanu, K. Driessens, G. Weiss, K. Tuyls, An automated measure of mdp similarity for transfer in reinforcement learning, in: Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence, 2014.
[37] J. Polastre, R. Szewczyk, D. Culler, Telos: enabling ultra-low power wireless research, in: IPSN 2005. Fourth International Symposium on Information Processing in Sensor Networks, 2005., 2005, pp. 364–369.
[38] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, et al., Tinyos: an operating system for sensor networks, Ambient Intell. 35 (2005) 115–148.
[39] G. Fortino, A. Guerrieri, G. O'Hare, A. Ruzzelli, A flexible building management framework based on wireless sensor and actuator networks, J. Netw. Comput. Appl. 35 (6) (2012) 1934–1952, doi:10.1016/j.jnca.2012.07.016.
[40] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, P. Levis, Collection tree protocol, in: Proceedings of the 7th ACM conference on embedded networked sensor systems, ACM, 2009, pp. 1–14.
[41] O. Gnawali, R. Fonseca, K. Jamieson, M. Kazandjieva, D. Moss, P. Levis, Ctp: an efficient, robust, and reliable collection tree protocol for wireless sensor networks, ACM Trans. Sensor Netw. (TOSN) 10 (1) (2013) 16.
[42] M. Johnson, M. Healy, P. van de Ven, M.J. Hayes, J. Nelson, T. Newe, E. Lewis, A comparative review of wireless sensor network mote technologies, in: SENSORS, 2009 IEEE, 2009, pp. 1439–1442, doi:10.1109/ICSENS.2009.5398442.
[43] C. Antonopoulos, A. Prayati, T. Stoyanova, C. Koulamas, G. Papadopoulos, Experimental evaluation of a wsn platform power consumption, in: 2009 IEEE International Symposium on Parallel Distributed Processing, 2009, pp. 1–8.