


## Article

# Heracles: A Context-Based Multisensor Sensor Data Fusion Algorithm for the Internet of Things

Flávia C. Delicato <sup>1,\*</sup> , Tayssa Vandelli <sup>2</sup>, Mario Bonicea <sup>2</sup> and Claudio M. de Farias <sup>2</sup>

<sup>1</sup> Department of Computer Science, Fluminense Federal University, Niteroi 24210-310, RJ, Brazil

<sup>2</sup> Tércio Pacitti Institute of Applied Computational Research, Federal University of Rio de Janeiro, Rio de Janeiro 21941-916, RJ, Brazil; tayssavandelli@gmail.com (T.V.); mario.boniceha@gmail.com (M.B.); cmicelifarias@cos.ufrj.br (C.M.d.F.)

\* Correspondence: fdelicato@gmail.com

Received: 17 September 2020; Accepted: 29 October 2020; Published: 4 November 2020



**Abstract:** In the Internet of Things (IoT), extending the average battery duration of devices is of paramount importance, since it promotes uptime without intervention in the environment, which can be undesirable or costly. In the IoT, the system's functionalities are distributed among devices that (i) collect, (ii) transmit and (iii) apply algorithms to process and analyze data. A widely adopted technique for increasing the lifetime of an IoT system is using data fusion on the devices that process and analyze data. There are already several works proposing data fusion algorithms for the context of wireless sensor networks and IoT. However, most of them consider that application requirements (such as the data sampling rate and the data range of the events of interest) are previously known, and the solutions are tailored for a single target application. In the context of a smart city, we envision that the IoT will provide a sensing and communication infrastructure to be shared by multiple applications, that will make use of this infrastructure in an opportunistic and dynamic way, with no previous knowledge about its requirements. In this work, we present *Heracles*, a new data fusion algorithm tailored to meet the demands of the IoT for smart cities. *Heracles* considers the context of the application, adapting to the features of the dataset to perform the data analysis. *Heracles* aims at minimizing data transmission to save energy while generating value-added information, which will serve as input for decision-making processes. Results of the performed evaluation show that *Heracles* is feasible, enhances the performance of decision methods and extends the system lifetime.

**Keywords:** data fusion; Internet of Things; smart environments

## 1. Introduction

The Internet of Things (IoT) [1] is the next generation of the Internet, which promises to endow potentially every physical object with sensors and/or actuators, allowing its integration between each of these physical objects and with other physical and virtual elements. Nowadays, the IoT has emerged from its infancy and evolved into a broad view encompassing the interconnection of sensors, actuators, the most varied intelligent objects and wireless sensor networks (WSNs). According to some authors, IoT is the backbone of a smart city [2] and WSNs are the sensing-actuation arm of the IoT. WSNs allow the seamless integration of sensing and communication elements into urban infrastructure, forming a digital skin over it [3]. The information generated by multiple WSNs in the IoT will be shared between different platforms, systems and applications, enabling an integrated and common view of the city. Such a view will provide inputs for well-grounded, timely and holistic decisions, in order to optimize processes and services, ultimately contributing to building cities as greener and more sustainable ecosystems.

The devices that make up the IoT have wireless communication capability and are equipped either with embedded sensors, as it is the case with smart phones, or with dedicated sensors, as in the case of WSNs [4]. Many IoT devices, mainly in the context of WSN, are powered by batteries. The wireless communication capabilities and the small dimensions of the devices in a WSN reduce the costs and complexity of implementing these networks and increase the flexibility and the resilience of the system as a whole. The low cost of the sensors means that, in general, there is redundancy in their deployment, so that, if some devices fail, the monitoring area will continue to be covered by the remaining devices. The downside of these devices is the dependency they create between the lifetime of the system and the duration of their batteries [5].

Average battery life is a relevant metric when evaluating an IoT system because it guarantees uptime without intervention in the environment, which can be undesirable or costly, depending on the purpose of the device and the application. In IoT, the system's functionalities are distributed among the devices that collect the data, those that transmit it and those that apply algorithms to process and analyze the data. A widely adopted technique for increasing the lifetime of an IoT system is to use data aggregation and/or fusion algorithms on the devices that process and analyze the data.

Algorithms for data aggregation and/or fusion executed within the network avoid unnecessary transmissions, thereby reducing the volume of data transmitted and energy consumption, making the network more efficient [6]. Data fusion techniques combine data from several sensors, aiming at achieving higher accuracy and more specific inferences than those obtained using a single sensor [6]. The outcome of a fusion operation can be a data synthesis or a transformation of the data from a state closer to the raw data to a level closer to a decision-making layer. In IoT, fusion algorithms aim at synthesizing the data, so that the volume of data transmitted is reduced, thus saving energy in the system [7].

There are already works in the literature proposing the application of data fusion techniques in the context of WSN and IoT systems [6,8–11]. However, until now, all works consider that the application requirements are previously known (at the design time), and the devices, algorithms and network behavior were designed specifically for the requirements of the target application (or applications). Such requirements vary from the nature of the sensed phenomenon, the sensing intervals and the data range to the events of interest. Such a feature was suitable for WSN systems, mainly in their first generation which, as mentioned, can be seen as intranets of sensing devices. However, with the IoT evolution, in current scenarios, it is not realistic to consider that such requirements are known a priori, particularly in the context of IoT for smart cities.

In a smart city, it is envisaged that the IoT will provide a sensing and communication infrastructure to be shared by multiple stakeholders, ranging from end users to corporate systems and government agencies. Applications that will make use of this infrastructure may not even exist at the time of its creation. In addition, to reach the full potential of an IoT ecosystem for smart cities, one must take advantage of the opportunistic nature of interactions, which make devices, services and applications appear and disappear at every moment, creating service opportunities in an ad hoc manner. Therefore, when designing a data fusion algorithm in this context, it should not be assumed that the information about the data nature, intervals or events that will be monitored is known, because the objective is to reuse the same infrastructure for different applications, in diverse contexts [7,12].

### Contributions

One of the first attempts to accommodate the discussed IoT requirements in terms of data fusion is described in [7]. In the paper, the authors propose *Hephaestus*, a data fusion algorithm that operates without the need to be aware of the application requirements. Instead, based on statistical properties of the data (kurtosis, asymmetry and mean), it separates the datasets into monomodal subsets. These subsets have the potential to map the events taking place in the monitored environment, without the need to have prior knowledge about them. Therefore, *Hephaestus* is able to identify and differentiate several events occurring in a monitored environment. However, it analyzes the data

based on descriptive statistics, and its limitation is the fact that different datasets may present the same asymmetry and, therefore, not be classified as different. In other words, the classification of events provided by *Hephaestus* is not perfectly accurate and may lead to the loss of relevant information or misinterpretation of phenomena occurring in the environment. By phenomena, we mean any occurrence of data masses (set of samples with a recognizable data peak) that may or may not correlate to events in the monitored environment. Our goal is to identify those phenomena and use them as an analysis basis to provide better input for an application's inferences.

In this work, we present a new algorithm called *Heracles*, which aims to overcome the limitations observed in algorithms based on descriptive statistics, such as *Hephaestus*. To improve the accuracy of event identification based on the analysis of the frequency concentration of the sensing data, we changed the perspective of analysis: Instead of using descriptive statistics, our approach adopts local maximums and minimums. The objective of such an approach is to prevent decisions regarding the grouping of data into subsets that denote events of relevance to any application from being made from just one number, as in *Hephaestus* (the outcome of its statistical analysis). Instead, mathematical functions are used with additional information, such as intervals of growth and decreases in values, in order to understand which are the local maximums and minimums.

*Heracles* does consider the context of the application, adapting to the dataset to perform the analysis. By context, we mean application features in terms of data thresholds, sensor interfaces and data density (number of samples in a given phenomenon). *Heracles* does not need to know such application requirements in advance, but its performance is directly linked to how adapted it is to the context and how adjusted its parameters are to the configuration of the monitored environment. For this reason, *Heracles* does not depend on the requirements of the applications, but on the monitored environment and its characteristics.

The final goal of our proposal is to contribute to providing an infrastructure for collecting and analyzing IoT data for smart cities, where multiple applications benefit, possibly in an ad hoc way, from the data generated and the information produced. The proposed fusion algorithm aims to, on the one hand, minimize data transmission to save energy from the sensing devices. On the other hand, it aims to generate value-added information of a higher quality than the raw data, which will serve as input to information systems and decision-making processes. By receiving a dataset and organizing it in different phenomena, *Heracles* will be able to provide a better input for inferences at higher decision levels. Such information will be generated agnostically to the requirements of specific applications, in order to be able to explore the wealth of data made available in IoT systems, in a creative and opportunistic way.

The remainder of this paper is organized as follows. In Section 2, we discuss related work. Section 3 presents our proposal in detail and Section 4 describes the experiments carried out to evaluate it. Section 5 provides final remarks and future research directions.

## 2. Related Work

With the challenge of taking advantage of the massive amount of data generated by sensor networks in the context of IoT systems, producing value-added information and knowledge has driven the publication of several works recently. In this section, we analyze some of these works, which are based on the use of different data analysis techniques. For each related work, we describe its general characteristics, its purpose and its similarities and differences in comparison to *Heracles*.

Examples of proposals with a similar purpose to *Heracles* are: [6,7,9–11,13–15].

In [9], the authors proposed a new average-based sensor fusion function that intends to fuse data of multiple applications simultaneously while handling uncertainty in the data samples. The authors argue that the sensor output should include a reading-related uncertainty value ( $\Delta y$ ), and the sensor observation should be an interval ( $y - \Delta y, y + \Delta y$ ). The reading interval is estimated by adding two tolerance values, respectively, to the left and the right of the pure value. The tolerance values may or may not be the same.

The following proposals deal with the challenge of executing data fusion algorithms for various applications in an IoT environment composed of large-scale wireless sensor networks.

The authors in [11] modified the moving average filter (MAF) technique to consider the importance of the sensed data for the application. The work of [11] presents an enhanced moving average filter, whose main idea is to weight the dataset to express the requirements (data ranges, data rates and states) of different applications. The disadvantage of this approach is the need for prior knowledge about the requirements of the application. As discussed, this premise is not always true in IoT scenarios, particularly in the smart cities domain, where new applications can be deployed in the ecosystem in an opportunistic or ad hoc way. In scenarios where the set of applications changes rapidly or frequently, EMAF is rendered unfeasible, needing to be constantly reconfigured. As EMAF [11] uses unprocessed data (readings with a low level of abstraction), high-level data of abstraction (such as decisions) are not properly handled. However, in later works, the same authors of EMAF presented high abstraction fusion methods [6]. In [6], the authors adapted some existing fusion methods to perform information fusion in data for multiple applications in the context of shared sensor networks. The authors proposed the following methods of information fusion: (i) enhanced Bayesian inference (EBI), (ii) enhanced Dempster–Shafer inference (EDSI) and (iii) enhanced fault tolerance average (EFTA).

EBI formalizes the combination of evidence according to the rules of Bayesian probability theory for each application [16]. It represents the hypothesis that the application “Y” will behave in a determined way given the result of the application “X”, considering the set of states of each isolated application. In EDSI, each application has its own set of hypotheses, which represents the application’s behavior. EDSI infers through the Dempster–Shafer inference method on the conditions of isolated applications and considers the premise that both applications go to a given state simultaneously. EFTA calculates the data intervals according to the requirements of each application, applying the traditional fault tolerance interval method. Then, EFTA produces the combination of each calculated interval.

Despite dealing with information for several applications, the solutions proposed in [6,11] present a significant drawback as they need to be pre-configured with the requirements of each application in the IoT network. While EMAF [11] needs to know the application requirements to properly weigh the dataset, EBI, EDSI and EFTA [6] need to infer the states for each application to present the decision that integrates various applications. Therefore, again the proposals fall in the limitation of not operating efficiently in scenarios where the set of applications in execution constantly changes. In contrast, in *Heracles*, we have introduced a new method of information fusion that does not depend on the requirements of the application, as long as the monitored environment is evaluated and used to calibrate the algorithm parameters.

In [13], the authors presented a novel decision-level data fusion technique for multiapplication WSNs (MWSN) based on data density which is able to handle decision conflicts. The proposed data fusion technique divides the dataset into groups based on application requirements and evaluates them separately. Each data group is known as an abstract sensor. The advantage of dividing the dataset into abstract sensors is to avoid interference among distinct data and better represent phenomena in the MWSN. Then, decisions are extracted from each abstract sensor and an analysis process is performed to verify if there are conflicts among the decisions. After that, a tableau is used to solve the conflicts excluding any undesired behavior. Different from [13], our work does not need to know a priori any application requirement. Moreover, our proposal has a different goal, since it is not a decision system but, instead, can be used jointly with a decision system to improve the quality of the decision outcome.

The authors in [14] proposed a distributed algorithm to detect phenomena, such as fires, oil spills (or spills of toxic gases) and others, in a monitored environment where the sensors are mobile devices and the phenomenon is dynamic (moving, growing or shrinking). The authors assume that the monitored environment does not have a central server to collect and aggregate data from the sensors. In the algorithm proposed in [14], sensors organize themselves into groups without intersection by choosing sensors as group leaders and assigning group members to sensors closer to the leader.

The sensors of each group send the data to the leader, which aggregates the collected data and detects the local phenomenon (in the area covered by the sensors of the group). Then, based on the order of the leaders' identifiers, each leader sends information on the detected local phenomenon to the next leader in that order. The receiving leader aggregates the information and sends it to the next leader, and so on. The last leader in the chain order aggregates information on all (locally) detected phenomena to discover the global phenomenon. In addition, the authors in [14] proposed two algorithms for electing leaders: Based on information from the global event and based on information from local phenomena. They adopt an optimization technique to reduce the energy costs of sending local information. The approach reduces the amount of data transmitted between leaders, since the information on local phenomena sent between them is summarized by a tuple of values represented by data limits, peaks and the volume of the set.

The proposal described in [14] has some similarities of purpose to our work as it is a distributed algorithm for event detection in a monitored environment. However, differently from our work, the authors in [14] assume the following premises: (i) the network sensors are homogeneous in terms of processing power, battery time, storage and communication range; and (ii) the sensors have knowledge of the "standard range of values" being monitored. The premise of the sensors being homogeneous is totally inadequate for an IoT environment, characterized by a high degree of heterogeneity of the devices. Our proposal has no such restriction. In *Heracles*, it is not necessary to specify the characteristics of the sensors or the data monitored at the time of system initialization. Our algorithm aims to adapt to the features of the monitored environment and network.

In [7], a distributed algorithm is presented that uses information fusion to aggregate sensor data. The proposal uses descriptive statistics to make a peak analysis and thus identify changes in the monitored environment. The proposed algorithm, called *Hephaestus*, calculates the mean, kurtosis and asymmetry of the input dataset. From these measurements, it decides whether or not to separate the set in two using the average as a parameter for dividing the sets. Then, it applies the same procedure to the subsets. *Hephaestus* shares several features with our proposal and its limitations served as motivation for the design of *Heracles*. In our current work, there is a better identification of the phenomena since we deal with the asymmetry to find the information peaks, which leads to a better characterization of the datasets in comparison with *Hephaestus*.

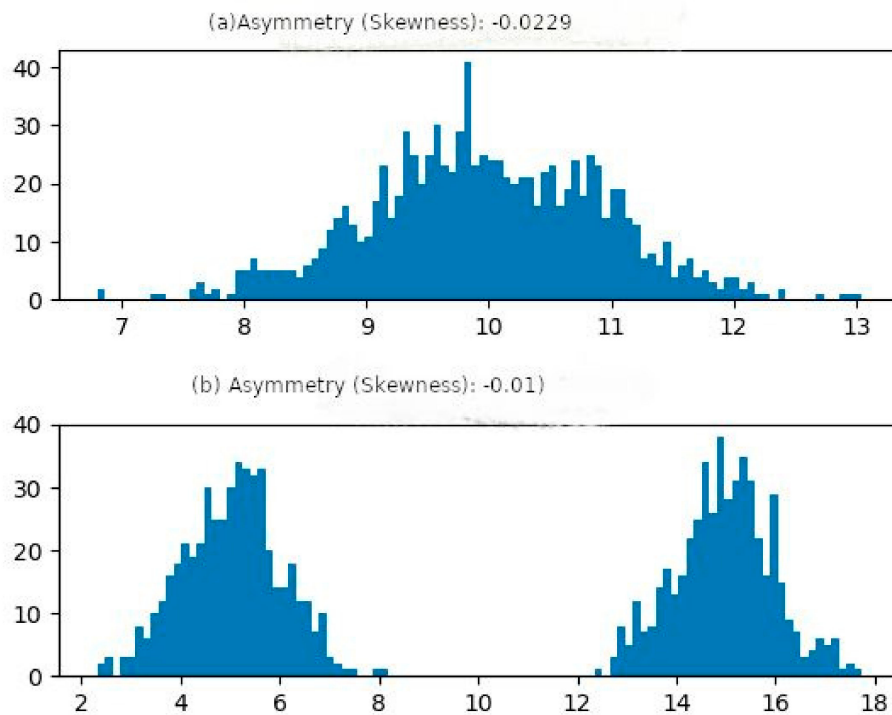
### 3. Heracles

In this section, we present *Heracles*, a context-based data fusion algorithm that does not consider the application requirements. This section is divided into: (i) an overview of the algorithm; (ii) the system model used; (iii) the phenomena model used; and, finally, (iv) the algorithm description.

#### 3.1. Overview

As mentioned, the proposal presented in [7], where the authors describe *Hephaestus*, served as an initial motivation for our work. The main limitation observed in *Hephaestus* is that its analysis is based on descriptive statistics (mean, kurtosis and asymmetry). With such an approach, different sets can present the same asymmetry and not be classified as being distinct (therefore representing different phenomena). As it can be seen in Figure 1, the two sets are strongly symmetric. When applying the *Hephaestus* approach, they will not be partitioned because the algorithm assumes that strongly symmetrical sets should not be further partitioned [17]. However, the second case, Figure 1b, presents two frequency peaks that will be considered as a single peak because they are strongly symmetrical, centered on 10. In this case, the ideal would be to partition this dataset into two subsets, each containing the data related to the described peaks.





**Figure 1.** Sets classified as platikurtic by Hephaestus—(a) monomodal symmetry (b) multimodal symmetry.

To work around such a limitation, which can lead to erroneous results of classification and identification of events, and therefore to inefficient decision making in IoT systems, we propose *Heracles*, which will be described in this section. *Heracles* is an algorithm aimed at characterizing phenomena through the use of data fusion. Its main goal is to carry out such a characterization of the phenomena while consuming few computational resources. The fact that *Heracles* is undemanding in terms of computational resources is extremely important to deal with IoT constrained devices. Even in the case of less restricted devices, the lower consumption of resources is a desirable feature to enable the sharing of computational capacities between multiple tasks.

*Heracles* considers that the nodes in the system can play one of two possible roles (detailed in Section 3.2) and the algorithm operation is divided into three main steps, namely: (i) data description; (ii) data filtering and (iii) data clustering. In the first step, the algorithm will simply analyze the current data, check if there is asymmetry on the collected samples and use descriptive statistics to summarize the data. In the second procedure, it will use the measurements computed in the previous procedure to identify potential peaks in the sampled data. Finally, in the last procedure, by using the identified peaks as an indication of break points on data, the algorithm will cluster the samples around those peaks into groups hereinafter known as phenomena. These phenomena will later be used as input to other systems or algorithms such as decision-level algorithms. All these three steps repeat at each data collection, according to a window.

In the following subsections, we describe the models used in this paper and provide a detailed explanation of the three steps of the algorithm.

### 3.2. System Model

The IoT network considered in this work includes several gateways and a set of heterogeneous sensor and actuator nodes that can belong to multiple physical networks. The gateway acts as a facade for gathering the latest information about the network execution context and sensor nodes status. The information includes, for each sensor node, its sensing capabilities, current residual energy, operation mode and geographical location. Such information is acquired by messages sent by sensor nodes whenever it is required from any gateway. Regarding the sensor nodes, they can play two roles:

(I) collector node and (II) fusion node. The collector node is responsible for collecting data and applying statistical methods to filter and describe data. The fusion node is responsible for collecting data from the sensor nodes, to identify different phenomena and send the output to the gateway. Such roles are determined a priori, at the deployment time, and depend on the device capabilities. Collector nodes are devices with less computational power than fusion nodes. The collector nodes are responsible for the description and data filtering steps of the algorithm, while the fusion nodes perform the data clustering. The collector nodes are responsible for realizing steps 1 and 2 sequentially, while the fusion node is responsible for performing step 3.

We model an IoT network as a forest  $F$  that comprises one or more physical networks. We model each physical network as an undirected graph  $G = (V, E)$ , where  $V = (v_1, v_2, \dots, v_n)$  represents the set of sensor nodes and  $E = (e_1, e_2, \dots, e_m)$  represents the set of all possible communication links among the sensor nodes in the same network.

For any given sensor node  $v_i$  in  $V$ ,  $i$  denotes the index of the node that belongs to the network. A sensor node  $V_i$  can provide one or more tasks depending on its capabilities to collect/sample different types of data, as, for instance, temperature, light, smoke and movement. Sensors can detect all events of interest occurring within their sensing range, provided that they have the required sensing unit. We assume that all the sensors in the physical network  $G_i$  have a valid communication path to reach at least one gateway. We also assume that the sensors in the same physical network are synchronized.

Each sensor may produce a data sample  $S_i$ . Let each sample of data be represented as a symbol. Each symbol is a tuple containing four values: sensorID, type, measurement and timestamp. SensorID is the sensor identification. It can be any unique identification, for example, a sensor MAC number or its ID in the network. Type is represented by an integer. Each integer denotes a given sensing unit type such as temperature and humidity. Measurement is a numerical value representing the data obtained by the sensor (the data sample). Finally, timestamp represents a timestamp of the moment when the data were collected. In this sense,  $S_2$ . measurement in our algorithm will represent a measurement produced by sensor 2.

### 3.3. Phenomena Model

For the execution of the proposed algorithm, we assume a scenario of a smart city where multiple sensors belonging to multiple sensor networks, which make up an IoT ecosystem, are producing data that will be consumed by different applications or decision-making systems. In this scenario, it is desired to take advantage of the data generated to obtain knowledge of a higher level than the raw data. Such knowledge will be potentially useful for several purposes, which are not known a priori. It is intended to explore the characteristic of dynamism and emerging behavior of an IoT ecosystem. This knowledge will be materialized in the form of a phenomenon. As nothing is known about the environment, a phenomenon represents a mapping on the data. Therefore, the output of *Heracles'* processing will be a set of phenomena identified from the analysis of the mass of data generated by the sensors. It is important to note that a phenomenon may or may not represent an actual real-world physical phenomenon. It is out of the scope of *Heracles* to map the identified phenomenon to actionable and fully meaningful information.

Each phenomenon  $P_i$  in a set of phenomena ( $P$ ) is represented by a tuple (TP, IR, FR, D, C). The type (TP) element denotes the type of monitored variable, such as temperature, humidity, current and several others. IR and FR variables mean initial range and final range. These variables denote the initial and final values of the phenomena data range. The scale of the variable will depend on the sensing type. D represents the phenomena density, that means the number of samples in this particular phenomenon. C represents the weighted value of these data. This is a way of representing the phenomena without having to store all data.

In order to discover the phenomena in the datasets, *Heracles* will have to store a vector with the relative frequency of each datum, meaning the number of times that a given sample appears divided by the number of samples in a fusion window (set of samples collected in a given time interval that will

be used by the algorithm); it will also use frequency measurement ranking (FMR), that is, an integer representing the average of this vector (a weighted average [11]). Finally, *Heracles* will have asymmetry of the fusion window (AS) as a structure to store the asymmetry value of the data samples. To ease the reader, we present a summary of the used data structure and acronyms in Table 1.

**Table 1.** Heracles data structures and acronyms.

| Data Structures and Acronyms Used in Heracles   |
|---|
| S—set of samples in the fusion window           |
| P—set of phenomena identified by Heracles       |
| TP—data type of the samples (e.g., temperature) |
| IR—initial range of the phenomena               |
| FR—final range of the phenomena                 |
| D—density of the phenomena                      |
| C—weighted value of the samples in a phenomenon |
| FMR—frequency measurement ranking               |
| AS—asymmetry of the fusion window               |
| PE—peak set                                     |
| AVG—weighted average                            |

The Heracles pseudocode is shown in Figure 2.

### 3.4. Data Description

In this phase, the proposed algorithm uses descriptive statistics methods to describe the sampled data (dataset within the defined fusion window). This procedure is performed periodically according to the duration of the fusion window. Two statistical measurements are computed in this phase: asymmetry and weighted average (lines 2–3). In order to calculate asymmetry, we refer to the Pearson's coefficient of skewness (Equation (3)) that uses the median of the dataset [18], as seen in Equation (1):

$$AS = 3(\text{mean}(\text{dataSet}) - \text{median}(\text{dataSet})) / \text{standardDeviation}(\text{dataSet}) \quad (1)$$

For calculating the weighted average, the procedure described in Figure 3 is adopted.

The computed measurements will later be used to cluster the data into phenomena. The asymmetry value will be particularly important since it indicates the possibility of having more than one phenomenon in the sampled data. If the sampled data are symmetric (in our case, we considered  $>1$  as an asymmetry value), the data will be clustered into a single phenomenon. In this case, traditional data fusion techniques can be used to process the data without any loss of accuracy. The weighted average will be used in the following phase of the algorithm—the data filtering.

### 3.5. Data Filtering Phase

In this phase, *Heracles* will filter data to identify possible peaks (samples with a higher occurrence in the dataset). First, *Heracles* will order the samples in the fusion window and store the frequencies of the samples (number of appearances in the fusion window). Then, it will consider any data above the value in FMR (as shown in Figure 3) as a potential peak. *Heracles* will store in a vector PE only the potential peaks and the number of samples until this peak (lines 7–9). It is important to notice that the data filtering phase is performed by each collector node. So, after the computation, the collector node will send to the fusion node the set of peaks, the weighted average stored in the FMR (as shown in Figure 2), the asymmetry value stored in AS and the size of S (line 13).



| <i>Heracles</i>              |   |
|------------------------------|---|
| <b>Input:</b>                | A set of Samples S  |
| <b>Output:</b>               | A set of phenomena Pi                                       |
| <b>For each sensing type</b> |   |
| 1.                           | <b># First step – Data Description (at collector nodes)</b> |
| 2.                           | Calculate the asymmetry of S and store on AS;               |
| 3.                           | Calculate the weighted average of S and store on FMR;       |
| 4.                           | <b># Second step – Data Filtering (at collector nodes)</b>  |
| 5.                           | Order the samples in S                                      |
| 6.                           | Store the frequencies of the samples in F                   |
| 7.                           | Freq = 0  |
| 8.                           | For i in S:   |
| 9.                           | Freq = Fi   |
| 10.                          | If Freq <= FMR  |
| 11.                          | Add (Si, freq) to Peak set PE                               |
| 12.                          | Freq = 0  |
| 13.                          | Send to Fusion node (PE, FMR, AS, size of S)                |
| 14.                          | <b># Third Phase – Data Clustering (at Fusion Node)</b>     |
| 15.                          | If AS < 1   |
| 16.                          | Then P = (IR of PE, FR of PE, Size of S, FMR)               |
| 17.                          | Else:   |
| 18.                          | <b>#search for phenomena</b>                                |
| 19.                          | IR = PEi  |
| 20.                          | FR = PEi+1  |
| 21.                          | For PEi, PEi+1 in PE:                                       |
| 22.                          | If the frequency of sample PEi > frequency of PEi+1:        |
| 23.                          | M = PEi   |
| 24.                          | If (PEi+1 > PEi) and (PEi+1 > M)                            |
| 25.                          | FR = PEi+1[0]   |
| 26.                          | AVG = Weighted Value (IR, FR)                               |
| 27.                          | SUM = number of samples between IF and FR                   |
| 28.                          | Add to P tuple (IR, FR, SUM, AVG)                           |
| 29.                          | IR = PEi+1[0]   |
| 30.                          | FR = PEi+2[0]   |
| 31.                          | M = 0   |
| 32.                          | Return P;   |

Figure 2. Pseudocode.

| Weighted Average |  |
|------------------|--|
| <b>Input:</b>    | A set of Samples S   |
| <b>Output:</b>   | A weighted Average AVG   |
| 1.               | Order the Samples S  |
| 2.               | Count the number of appearances of each Sample and Store in W (samples, appearances) |
| 3.               | For $W_i$ in W   |
| 4.               | $AVG = W_i[0] * W_i[1] / \text{Size}(S)$   |
| 5.               | Return AVG   |

**Figure 3.** Average pseudocode.

Besides dealing with the resource constraints of sensor nodes, by filtering the data in the fusion window, the algorithm is also able to further reduce the data transmitted. By using the FMR, it is possible to give priority to the peaks in the fusion window, and in this sense, *Heracles* will be able to better observe the existence of different phenomena in the data samples. The rationale behind this step is that not all data are relevant to identify the phenomena or not all data will be useful to make a decision about some situation. So, in order to reduce data transmission, we will only send potential peaks (those data above the FMR). The purpose of this step is to highlight the data that will be useful to identify some potential phenomena.

### 3.6. Clustering Phase

This phase is optional, and its execution depends on the characteristic of the analyzed data. After the data filtering phase, the algorithm will evaluate if there are (potentially) multiple phenomena or a single phenomenon in the monitored region. If the dataset is symmetrical, then there is only potentially one mapped phenomenon (line 16), so this is an indication that these data can be fused by a traditional decision-level data fusion algorithm. On the other hand, if the dataset has an asymmetry value above 1, then the algorithm decides that a clustering routine must be performed in order to separate the different phenomena found in the samples (line 18). Given that the data in the fusion window are asymmetric, *Heracles* will search for the data sample with more occurrences in the window, and this value will be set as the initial range of a phenomenon (lines 19–23). Then, the algorithm will search in the fusion window for the next value that is lower than the previous peak (line 24) and will set this point as the final range. Following, the number of samples between the initial and final range will be counted, and this number will be the density value (line 27). Finally, the average of the samples between the initial and final range values will be calculated. This is characterized as a phenomenon. All these steps will be repeated until the end of the fusion window. For each phenomenon, the values of initial range, final range, density and average will be stored. This is a simple way to represent the phenomena and reduces the amount of data to be transmitted. The output of *Heracles* will be a set of phenomena that will be used by some decision-level algorithm to infer about the monitored environment. It is noteworthy to state that a phenomenon may or may not have correspondence to a given physical event. This set of phenomena can later be used by other decision-level techniques. As previously discussed similarly to Hephaestus, *Heracles'* goal is to provide a set of phenomena to be used by a decision-level fusion method, improving its performance.

## 4. Experiments

In this section, we describe the experiments performed for assessing *Heracles* regarding different aspects. We defined two main goals for the evaluations. The first goal is to analyze *Heracles'* overhead in terms of communication and energy consumption. The second goal is to analyze the accuracy in terms of its success in correctly inferring the phenomena used in our case study.

To assess *Heracles'* overhead in terms of communication and energy consumption of the IoT network, we compare it with: (i) the naïve Bayes (NB), a well-known decision-level data fusion algorithm; (ii) the Hephaestus algorithm used as input to the NB; and (iii) the density-based data fusion algorithm presented in [13].

To assess accuracy (in terms of success in correctly inferring the phenomena), we compare *Heracles* with the naïve Bayes (NB) for multiple applications, but without being aware of the requirements of the applications. We also compare *Heracles* with the proposal described in [13], which is a multiapplication algorithm, and with Hephaestus, also without being aware of the requirements of the applications.

In this section, we first describe the adopted scenario, implementation details and the metrics used in our experiments. Thereafter, we discuss the results obtained in the performed experiments. All experiments lasted for one hour and were repeated 30 times, and the presented results had a confidence interval of 95%.

#### 4.1. Scenario

A smart city is an application domain that encompasses several other domains, such as smart buildings, smart homes and smart energy, to name a few. We chose two applications for testing our proposed algorithm: overhead power line monitoring and battery monitoring. They are important applications in the smart grid domain, which is part of the smart city umbrella of domains. A smart grid is a type of bidirectional electrical grid where sensors gather information from both vendors and consumers in order to reduce energy waste and improve the grid efficiency. The overhead power line monitoring application has great importance because in the last few years the loads of the power lines have been increasing. Knowledge about the power line temperature is paramount for making decisions about its loading. Due to the increasing load of the power lines, more electric energy turns into heat. This heating could damage the lines, causing operational failures, which could lead to blackouts. The usual and safe line temperature is around 40–65 °C [18,19]. In the battery monitoring application, batteries are used as a solution to solve the problem of unexpected peaks in the electricity demand. Overheating could damage the battery; therefore, the temperature must be monitored. High temperature values could also be evidence of energy waste [20] (electricity turning into heat). The usual battery temperature is in the interval from 40 to 144 °C.

#### 4.2. Implementation and Environmental Configuration

*Heracles* algorithm was implemented in python programming language and the implemented code was deployed in both fusion and collector nodes (according to their respective functionalities). As the hardware platform for the sensor nodes, the ESP8266 board was adopted. ESP8266 is a microcontroller board widely used in IoT applications. It has a Wi-Fi shield that allows connecting to wireless networks using TCP/IP. It supports different languages, including Micropython—A software implementation of the programming language based on Python 3.0, written in C, optimized for microcontrollers. Therefore, for the embedded software implementing the *Heracles* algorithm in real nodes, we have used Micropython. For the simulation (performed in order to evaluate scalability), we have used the Simpy framework, which is a well-known simulation library. As the protocol stack for communication, we have used CoAP in the application layer, UDP in the transport layer, 6LoWPAN and 802.11.

In all performed experiments (except the scalability experiment), the network was composed of 23 sensor nodes randomly divided into two networks (16 sensors are collector nodes (CN), 5 are fusion nodes (FN) and 2 are gateways (GW)). The nodes were arranged in a flat and static network topology. The nodes can use accelerometers, temperature, light, humidity and presence sensing units. All experiments were performed in an environment consisting of a 50 × 50 m field. The network nodes were randomly arranged over a squared area defined as {(0,0), (50,0), (0, 50), (50, 50)}. To verify the impact of monitoring and transmitting over a long-range area, the gateways are located far from any sensor nodes, at coordinates (200, 100) and (100, 200). All gateways are connected to at least one fusion node. All sensor nodes start with 0.5 joules as initial energy within their batteries [7].

### 4.3. Metrics

The metrics used in the performed experiments for evaluating resource consumption were the memory and the energy consumption. The memory consumption is defined as the amount of memory used by our algorithm installed in the nodes (RAM and ROM) and the energy consumption is defined as the amount of energy consumed by the network, i.e., the total amount of energy consumed by each node when executing the steps of the algorithm. To evaluate the energy consumed by the execution of *Heracles*, we used the energy model used in [7]. Overall, the energy consumption of a sensor node during time  $t$  is calculated as shown in Equation (2):

$$E(T) = E_c + E_s \quad (2)$$

where  $E_c$  represents the energy consumption of the communication module, and  $E_s$  represents the energy consumption of the sensing module. For the ease of analysis, we assume that the data exchange between two neighboring sensor nodes (within the one-hop communication range) belonging to the same network is performed through direct communication. The energy consumption of transmitting 1-bit (message of 1 bit in size) data over distance  $d$  is defined as  $E_{tx}(l, d)$  [6], computed as shown in Equation (3):

$$E_{tx}(l, d) = E_{elec} \times l + \epsilon_{amp} \times l \times d^2 \quad (3)$$

where  $E_{elec}$  and  $\epsilon_{amp}$  are hardware-related parameters [6]. We also assume that the receiver does not consume energy in the data exchange process. For any two distance sensors (outside the one-hop communication range, still belonging to the same network), the data communication is transferred by using the shortest path-based multihop routing protocol (please note that the routing process is out of the scope of our work). The energy consumption of transmitting 1-bit (message of 1 bit in size) data from the source (src) to the destination (des) is defined as  $E_{tx}(l, src, des)$ , as shown in Equation (4):

$$E_{tx}(l, src, des) = \sum_i k E_{tx}(l, d) \quad (4)$$

where  $d$  is replaced from hop to hop,  $i$  is an iterator and  $k$  is the minimum hop count that the data travel from source to destination. The energy consumption of the sensing module is calculated by the linear Equation (5):

$$E_{si} = E_{Rsi} \times t_{si} \quad (5)$$

where  $E_{Rsi}$  represents the energy consumption of service  $i$  in one time unit and  $t_{si}$  represents the time period for performing service  $i$ .

The metric used for accuracy was Matthew's correlation coefficient (MCC). Given the true positive rate TP (percentage of correctly classified as true occurrences of a given event), true negative rate TN (the event did not occur and this fact was correctly identified), false positive rate FP (percentage of instances classified as true but actually false) and false negative rate FN (percentage of instances classified as false but actually true), the MCC is computed as shown in Equation (6):

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \quad (6)$$

MCC obtains the maximum score (+1) when both true positive and true negative are 100\% and the minimum score (−1) when false positive and false negative are 100\%. MCC is thus a good measure to strike a balance between the accuracies of multiple classes.

Regarding FP, FN, VP and VN, we must remember that the outputs of *Heracles* are mapped phenomena, as stated in the previous sections. The goal of the evaluation in terms of accuracy is to assess if by using *Heracles* we can improve the decision-making process. A decision-making process is as good as the quality and scope of its inputs, and in this case, the inputs are the potential phenomena occurring in the environment. Therefore, what is going to be actually evaluated is whether the set *Heracles* + naïve Bayes is better than only using naïve Bayes or using Hephaestus + naïve Bayes.

We claim that by providing the identified phenomena as inputs to the naïve Bayes, we will achieve a more accurate decision to detect when an application changed its state (from a healthy state to a failure) since the phenomena will map those states in the fusion window. An example of mapped phenomena considered in our scenario, specified according to the adopted phenomena model (see Section 3.3), is (Temperature, 55, 72, 23, 63.7). This tuple means that the temperature samples varied from 55 to 72, there were 23 samples in the window and an average value of 63.7 °C—what could be later mapped by a decision system as fit for the OLPM application.

In our case, for the set *Heracles* + naïve Bayes, FP indicates when a mapped phenomenon will make the decision system believe (and act upon such a decision) that an application changed its state, when it should not have changed, for example, an OPLM application that detects a transmission failure when there is no transmission failure. FN indicates when a mapped phenomenon will make the decision system believe that an application has not changed state when it should have changed, for instance, an OPLM application that does not detect a transmission failure when the temperature reaches a given threshold that denotes a failure. TP indicates a situation when a mapped phenomenon will make the decision system believe that an application has changed states correctly (a transmission failure is detected, and it really occurred, for example) and TN indicates when a phenomenon indicates to the decision system that an application has maintained its state correctly (an OPLM does not detect a transmission failure fire when there is no transmission failure, for example). Considering the example of the phenomena mentioned above (Temperature, 55, 72, 23, 63.7), if this is used as an input for a naïve Bayes classification (what is the probability that the overhead power line is faulty given that the temperature mean is 63.7?), the answer is going to be not faulty. In this sense, *Heracles* can clean and reduce data sent to decision systems. On the other hand, if we consider the phenomena (Temperature, 75, 102, 16, 93.7), the result will be that the overhead power line is faulty.

#### 4.4. Evaluating Memory Consumption

A metric chosen to analyze the feasibility of deploying *Heracles* in real nodes and its overhead is RAM and ROM consumption. Regarding ROM memory, *Heracles* occupied around 48 Kb of ROM, making it feasible in the ESP8266. For evaluating RAM consumption, we separated some input configurations to be able to measure the impact that the fusion window size may or may not have on RAM consumption. We have used the library called Psutil (<https://pypi.org/project/psutil>) to help us collect the desired amount of RAM consumed.

To properly evaluate RAM consumption, we have varied the size of the fusion windows during the algorithm operation. The fusion window size consisted of an entry with 50 data samples only. We obtained an average of 45.2 Kb of RAM  $\pm$  3.2 kb of RAM used during the execution of the algorithm. For the second configuration, we designated a 200-data samples entry. We obtained an average of 49.2 Kb  $\pm$  4.6 kb of RAM used. Even increasing the fusion window size to 1000 samples, RAM consumption on average was 51.9 Kb  $\pm$  4.8 kb. Since ESP8266 has 64 kb of RAM, we can conclude that it is feasible to deploy *Heracles* in an ESP8266 board, a suitable piece of hardware for IoT devices.

#### 4.5. Evaluating Energy Consumption

The nodes started with 0.5 J of initial energy, similarly to the initial energy used in [7]. In Figure 4, the results for assessing the network lifetime (measured as the first node to die in the network) are shown. In Figure 4, we can observe that the network lifetime (for 2, 4, 6, 8 and 10 phenomena) using (i) *Heracles*, (ii) Hephaestus and (iii) [13] is greater than the network lifetime using only naïve Bayes. We can also observe that the network lifetime when using *Heracles* is greater (9.8% better) than using only Hephaestus and 12.3% better than using the proposal described in [13]. There are two major facts that explain the obtained results for the system lifetime: (i) the number of messages transmitted by *Heracles* does not increase as more phenomena are found in the network; (ii) the size of the messages transmitted by *Heracles* and Hephaestus increases as more phenomena are found in the network and

(iii) since [13] uses the application requirements in the decision process, as the number of phenomena increases, energy consumption also increases since the algorithm has to analyze more information.

## System Lifetime

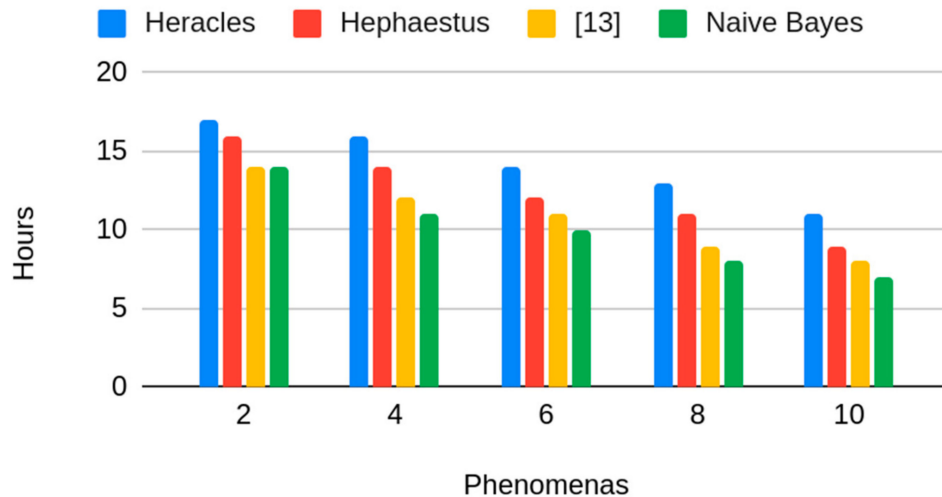


Figure 4. Lifetime.

### 4.6. Evaluating Accuracy

This section describes the simulated experiments for evaluating the accuracy achieved using *Heracles* compared to Hephaestus, [13] and naïve Bayes in terms of their ability to correctly identify the different events of interest for the smart city applications described in our case study.

Since neither Hephaestus nor *Heracles* are decision-level algorithms, we intend to use their discovered phenomena as input for the naïve Bayes (NB) decision algorithm. We performed four simulated experiments: (i) the first uses *Heracles*, (ii) the second uses Hephaestus, (iii) in the third experiment, we used only naïve Bayes and, finally, in (iv), we used the density-based decision-making data fusion algorithm presented in [13]. In all simulated experiments, it was not necessary to be aware of the requirements of the deployed applications. The size of the fusion window for *Heracles*, Hephaestus, the algorithm proposed in [13] and naïve Bayes was set as 200 samples. This set of experiments intends to show how effectively *Heracles* can improve decision systems by providing better decision inputs.

During these experiments, there were four time slots (T1, T2, T3 and T4). So, for each time slot, *Heracles*, Hephaestus, [13] and NB were tested for comparing the achieved results in terms of MCC values. Table 2 shows the achieved results. The first row represents *Heracles*' MCC score. The second row shows Hephaestus' performance. The third row presents NB's performance.

Table 2. Experiments.

|    | <i>Heracles</i> | Hephaestus      | NB            | [13]            |
|----|-----------------|-----------------|---------------|-----------------|
| T1 | 0.9988 ± 0.0012 | 0.9966 ± 0.0011 | 0.9432 ± 0.06 | 0.9988 ± 0.0103 |
| T2 | 0.9976 ± 0.0014 | 0.9926 ± 0.0023 | 0.946 ± 0.28  | 0.9958 ± 0.0034 |
| T3 | 0.9847 ± 0.009  | 0.9588 ± 0.011  | 0.901 ± 0.25  | 0.9722 ± 0.033  |
| T4 | 0.9269 ± 0.014  | 0.8464 ± 0.018  | 0.883 ± 0.12  | 0.906 ± 0.02    |

For the T1 time slot, both applications are in an ideal condition. In this case, given the unequal data ranges, the composition of both applications generated a highly skewed dataset. Hephaestus and [13] divided the dataset in two distinct groups (through different strategies), identifying the different datasets, and applying NB in each set. The NB, on the other hand, grouped all the data into a



single group. *Heracles* was also able to divide into distinct groups (our phenomena), which lead to a better performance than NB. *Heracles*, Hephaestus and [13] had similar results in this time slot (since they could divide data into similar results). Applying NB directly to the dataset biased the fusion result. Such bias is responsible for the worst accuracy of NB (94.32%) in comparison to Hephaestus, the density-based algorithm [13] (99.88%) and *Heracles*' accuracy (99.66%).

For the T2 time slot, the OPLM app was in an unhealthy condition, and both applications overlapped, which produced an almost symmetric dataset. *Heracles*, [13] and Hephaestus recognized this dataset as a single dataset, but with a slightly shifted mean over the more concentrated data range. The behavior of this mass of data indicates a concentration over the overhead power line-generated data which is overloaded. The shifted mean over the overhead power line-generated data (that were mapped into a single phenomenon) was responsible for the greater accuracy of *Heracles*, [13] (99.58%) and Hephaestus (99.26%) over NB (94.6%).

For the T3 time slot, the BM was in an unhealthy condition, and a highly asymmetric dataset was formed by the composition of the different applications. Due to the high asymmetry in the dataset, *Heracles* and Hephaestus were capable of identifying that there were different applications in the monitored area. However, *Heracles* and [13] were better at representing and separating the phenomena, thus leading to a better result (there were clearly three phenomena mapped as exemplified by Figure 5—A more accurate representation of the environment). NB weighted all data equally, leading to the worse performance of the three (90.1%).

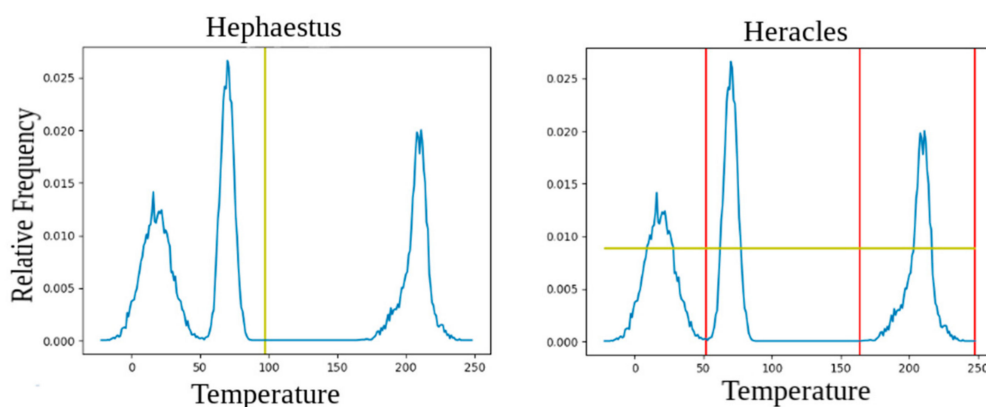


Figure 5. Comparing *Heracles*' and Hephaestus' phenomena discovery.

By comparing the accuracy achieved by *Heracles* and Hephaestus at the T4 time slot, where both applications were in unhealthy conditions, with the accuracy at other time slots, we can notice a fall in the Hephaestus result, while *Heracles* and [13] still have an upper hand. The dataset obtained at the T4 time slot is an almost symmetric multimodal dataset. Due to its symmetry, depending on the samples Hephaestus receives as input, the dataset is recognized as a symmetric dataset, which does not need to be divided, producing a single dataset. Meanwhile, *Heracles* searches for asymmetry and peaks that make the phenomena more identifiable. As T4 presents an almost symmetric dataset, the accuracy of Hephaestus is lower (84.64%) than the accuracy obtained at the other time slots (99.66% for T1, 99.26% for T2 and 95.88% for T3), while *Heracles* is almost unaffected. We can see that by consistently mapping the phenomena using the peak and asymmetry strategy, *Heracles* was able to deliver a better overall performance in all experiments.

#### Effect of the Fusion Window Size on T4

Considering that the T4 time slot is the most challenging one due to the symmetry of its data, we decided to further investigate how much the sample size interferes on the accuracy of *Heracles*. For this experiment, we ran *Heracles* on the T4 time slot analyzing 200, 400, 600 and 800 samples.

The results in Table 3 reflect that as *Heracles* analyzes more samples, its accuracy increases. This result is expected, since as the sample size increases, the observation of the real world is more complete and the associated uncertainty is lower. This result is important since it presents a solution to a situation where *Heracles* has a distinctive better performance over Hephaestus, since the latter is unable to correctly recognize a dataset due to a particular symmetric configuration.

**Table 3.** Window size vs. accuracy.

| Sample Size | Accuracy on T4        |
|-------------|-----------------------|
| 200         | $0.9269 \pm 0.018099$ |
| 400         | $0.9645 \pm 0.007517$ |
| 600         | $0.9812 \pm 0.004674$ |
| 800         | $0.9967 \pm 0.002032$ |

In Figure 5, we can see that while Hephaestus only separates the phenomena from the calculated average, *Heracles* considers that there is an advantage when looking specifically for events to separate peaks. This case exemplifies how the separation of the main event (peak centered at 65) can be advantageous in *Heracles*, while Hephaestus characterizes the two separate regions as sufficiently symmetrical.

#### 4.7. Evaluating Scalability

This section describes the experiments conducted for evaluating *Heracles'* scalability—If the algorithm proposed is still feasible as the number of phenomena in the monitored environment increases. This is extremely important in smart city scenarios, where numerous phenomena of interest are possibly occurring simultaneously and can serve as input for different information systems and applications. For this set of experiments, we have performed experiments using a total of 20, 40, 60, 80 and 100 phenomena. We set the fusion window with 200 samples. For each phenomenon, we randomly assigned one sensing unit from a set of five different sensing units (accelerometers, temperature, light, humidity and presence) with randomly assigned sensing rates varying from 1 to 5 s, using the procedures explained in [21]. It is discussed in the literature that random monitoring tasks may not always represent real applications; however, the diversity they provide is sufficient for this group of experiments as explained in [21]. Each phenomenon had an interval, randomly generated, varying from 0 to 1000 (changing the measure unit according to the sensing type).

Regarding accuracy, we can see in Figure 6 that the accuracy results are almost not affected by the number of phenomena, since *Heracles* is agnostic of the number of phenomena. This shows that *Heracles* scales well in terms of the provided accuracy. Regarding system lifetime, since the number of phenomena influences in the size of the message transmitted to the fusion node, we can observe in Figure 7 that the system lifetime decreases as the number of phenomena increases. It is worth noticing that since [13] is an application-aware algorithm, as the number of applications increases, its system lifetime is the most reduced. We also can see that this decrease is not linear, meaning that *Heracles* is able to extend the system lifetime even as the number scales.

## Scalability - Accuracy

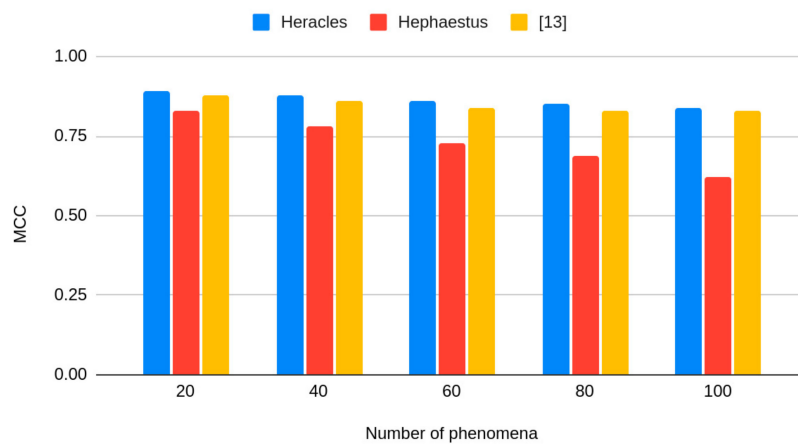


Figure 6. Accuracy scalability.

## System Lifetime

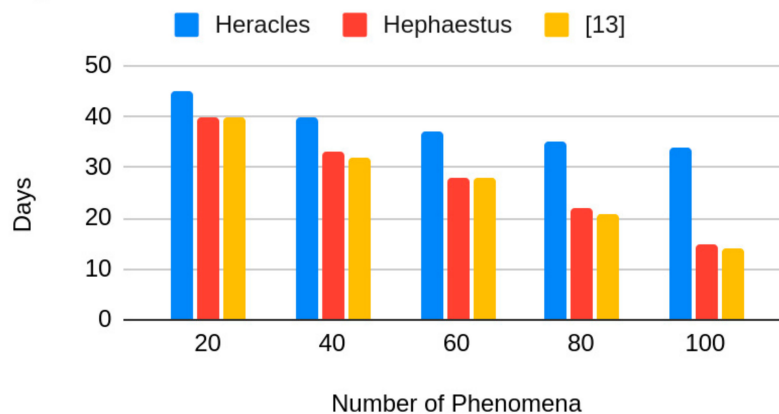


Figure 7. System lifetime scalability.

## 5. Conclusions

As the amount of data generated by physical objects increases, the IoT becomes an important asset to monitor real-world phenomena and a rich source of data for emerging applications. However, in order to have effective value for the applications, the raw data produced by instrumented objects must pass through a whole transformation chain to generate insights for decision making and high-level actionable knowledge. *Heracles* is an algorithm aimed at contributing to such a complex process of data transformation and knowledge production. In the same way as a previously proposed algorithm called *Hephaestus* [7], *Heracles* transforms the monitoring environment into a data repository to be consumed by applications and information systems. Further, as with *Hephaestus*, instead of delivering the collected raw data to such systems and applications, *Heracles* adds value to such data, improving them with insights about the monitored environment, taking a first step towards the production of useful and actionable knowledge. However, unlike *Hephaestus*, *Heracles* is able to cluster phenomena based on peaks which leads to a better mapping of symmetric phenomena (such as the one presented in T4). This gives *Heracles* a better accuracy. Moreover, due to an adopted filter, *Heracles* can extend the system lifetime even when the number of phenomena increases. This is an important feature in smart city scenarios, where there will be an enormous number of applications opportunistically making use of the implanted sensors and consuming their resources.

Although simple, we believe that *Heracles* has the potential to become a valuable support tool to be used along with more sophisticated techniques in the quest to build a value chain from the vastness

of data produced in the IoT. In future works, we intend to: (i) develop a decision-level fusion algorithm that is agnostic to the application requirements; (ii) provide a tinyML approach (such as reinforcement learning) to better cluster the data; and (iii) correlate phenomena from different data types to better map things happening in the real world.

**Author Contributions:** Conceptualization, F.C.D., C.M.d.F., T.V. and M.B.; methodology, C.M.d.F., T.V. and M.B.; software, T.V. and M.B.; validation, F.C.D., C.M.d.F., T.V. and M.B.; investigation, F.C.D., C.M.d.F., T.V. and M.B.; resources, F.C.D. and C.M.d.F.; data curation, T.V. and M.B.; writing—original draft preparation, F.C.D., C.M.d.F., T.V. and M.B.; writing—review and editing, F.C.D. and C.M.d.F.; supervision, F.C.D.; project administration, C.M.d.F. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was partially funded by FAPESP, grant number 2015/24144-7. Flavia Delicato is a CNPq Fellow.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

- Atzori, L.; Iera, A.; Morabito, G. The Internet of Things: A survey. *Comput. Netw.* **2010**, *54*, 2787–2805. [\[CrossRef\]](#)
- Chourabi, H.; Nam, T.; Walker, S.; Gil-Garcia, J.R.; Mellouli, S.; Nahon, K.; Scholl, H.J. Understanding smart cities: An integrative framework. In Proceedings of the IEEE 2012 45th Hawaii International Conference on System Sciences, Maui, HI, USA, 4–7 January 2012; pp. 2289–2297.
- Dameri, R.P. Searching for Smart City definition: A comprehensive proposal. *Int. J. Comput. Technol.* **2013**, *11*, 2544–2551. [\[CrossRef\]](#)
- Jin, J.; Gubbi, J.; Marusic, S.; Palaniswami, M. An information framework for creating a smart city through internet of things. *IEEE Internet Things J.* **2014**, *1*, 112–121. [\[CrossRef\]](#)
- Kopetz, H. *Real-Time Systems: Design Principles for Distributed Embedded Applications*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2011.
- Gubbi, J.; Buyya, R.; Marusic, S.; Palaniswami, M. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Gener. Comput. Syst.* **2013**, *29*, 1645–1660. [\[CrossRef\]](#)
- Farias, C.; Pirmez, L.; Delicato, F.; Carmo, L.; Li, W.; Zomaya, A.Y.; de Souza, J.N. Multisensor data fusion in shared sensor and actuator networks. In Proceedings of the 17th IEEE International Conference on Information Fusion (FUSION), Salamanca, Spain, 7–10 July 2014; pp. 1–8.
- Aquino, G.; Pirmez, L.; de Farias, C.M.; Delicato, F.C.; Pires, P.F. Hephaestus: A multisensor data fusion algorithm for multiple applications on wireless sensor networks. In Proceedings of the IEEE 2016 19th International Conference on Information Fusion (FUSION), Heidelberg, Germany, 5–8 July 2016; pp. 59–66.
- Wagner, A.; Speiser, S.; Harth, A. Semantic web technologies for a smart energy grid: Requirements and challenges. In Proceedings of the 9th International Semantic Web Conference (ISWC2010), Bonn, Germany, 11 November 2010; pp. 33–37.
- Singhal, D.; Garimella, R.M. Simple Median based information fusion in wireless sensor network. In Proceedings of the IEEE 2012 International Conference on Computer Communication and Informatics, Coimbatore, India, 10–12 January 2012; pp. 1–7.
- Dziengel, N.; Seiffert, M.; Ziegert, M.; Adler, S.; Pfeiffer, S.; Schiller, J. Deployment and evaluation of a fully applicable distributed event detection system in Wireless Sensor Networks. *Ad Hoc Netw.* **2016**, *37*, 160–182. [\[CrossRef\]](#)
- De Farias, C.M.; Pirmez, L.; Delicato, F.C.; Dos Santos, I.L.; Zomaya, A.Y. Information fusion techniques applied to Shared Sensor and Actuator Networks. In Proceedings of the 37th Annual IEEE Conference on Local Computer Networks IEEE, Clearwater, FL, USA, 22–25 October 2012; pp. 188–191.
- Zaslavsky, A.; Perera, C.; Georgakopoulos, D. Sensing as a service and big data. *arXiv* **2013**, arXiv:1301.0159.
- De Farias, C.; Delicato, F.; Fortino, G. *A Density-Based Decision-Making Data Fusion Method for Multiapplication Wireless Sensor Networks*; IEEE: Fukuoka, Japan, 2019; pp. 840–847.
- Safia, A.A.; Al Aghbari, Z.; Kamel, I. Phenomena detection in mobile wireless sensor networks. *J. Netw. Syst. Manag.* **2016**, *24*, 92–115. [\[CrossRef\]](#)
- Kumar, S.; Chaurasiya, V.K. A multisensor data fusion strategy for path selection in Internet-of-Things oriented wireless sensor network (WSN). *Concurr. Comput. Pract. Exp.* **2018**, *30*, e4477. [\[CrossRef\]](#)

17. Bayes, T. An essay towards solving a problem in the doctrine of chances. 1763. *Md Comput. Comput. Med. Pract.* **1991**, *8*, 157.
18. Sanquetta, C.R.; Behling, A.; Dalla Corte, A.P.; Netto, S.P.; Rodrigues, A.L.; Simon, A.A. A model based on environmental factors for diameter distribution in black wattle in Brazil. *PLoS ONE* **2014**, *9*, e100093. [[CrossRef](#)] [[PubMed](#)]
19. Denholm, P.; Kulcinski, G.L.; Holloway, T. Emissions and energy efficiency assessment of baseload wind energy systems. *Environ. Sci. Technol.* **2005**, *39*, 1903–1911. [[CrossRef](#)] [[PubMed](#)]
20. Raji AR, O.; Villegas Salvatierra, R.; Kim, N.D.; Fan, X.; Li, Y.; Silva, G.A.; Tour, J.M. Lithium batteries with nearly maximum metal storage. *ACS Nano* **2017**, *11*, 6362–6369. [[CrossRef](#)] [[PubMed](#)]
21. Boughorbel, S.; Jarray, F.; El-Anbari, M. Optimal classifier for imbalanced data using Matthews Correlation Coefficient metric. *PLoS ONE* **2017**, *12*, e0177678. [[CrossRef](#)] [[PubMed](#)]

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).