Computer Communications 176 (2021) 190–206

Contents lists available at ScienceDirect

Computer Communications

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

On a multisensor knowledge fusion heuristic for the Internet of Things

Gabriel Martins [a,*], Sergio Guedes de Souza [b], Igor Leão dos Santos [d], Luci Pirmez [c], Claudio M. de Farias [a]

[a] PESC, Federal University of Rio de Janeiro - UFRJ, Brazil
[b] NCE, Federal University of Rio de Janeiro - UFRJ, Brazil
[c] PPGI, Federal University of Rio de Janeiro - UFRJ, Brazil
[d] PPPRO, Federal Centre for Engineering Studies and Technological Education – CEFET/RJ, Brazil

ARTICLE INFO

Keywords:
Multisensor knowledge fusion
Multisensor data fusion
Wireless sensor networks
Internet of things
Knowledge management

ABSTRACT

Internet of Things (IoT) is envisioned as the interconnection of the Internet with sensing and actuating devices. IoT systems are usually designed to collect massive amounts of data from multiple and possibly conflicting sources. Nevertheless, data must be refined before being stored in a repository, so as information can be correctly extracted for further uses. Knowledge fusion is an important technique to identify and eliminate erroneous data from compromised sources or any mistakes that might have occurred during the extraction process. We propose a new multisensor knowledge fusion heuristic (MKFH) for IoT supporting the knowledge extraction and transfer needed to further knowledge management, also discuss the role of reinforcement learning over integration on a multi-application wireless sensor/actuator network (WSAN). Results shows that the proposed multisensor knowledge fusion heuristic is compatible with the IoT paradigm and enhances integration.

1. Introduction

The Internet of things (IoT) environments are composed of ordinary things attached to computational devices from which information regarding the thing and its environment may be gathered, shared, and consumed by information systems to allow smart services [1]. Indeed, some IoT environments are not only resource-limited but also produce data in low semantic levels. A vital challenge from the IoT paradigm is to permit the discovery, access, integration, and analysis of knowledge to enable appropriate entity linkage. Once converted into information, data may be used to make better decisions and improve management. A reasoning process may trigger preconceived actions by assessing the environment to add meaning to its behaviour [2,3]. It may identify more optimal strategies by selectively applying preconceived premises (knowledge) to assesses a situation [4]. Thus, the ability to extract and transfer knowledge through a network is essential to enhance decision making. A classical approach for knowledge management process states that data generates information, information produces knowledge, and knowledge yields wisdom [4,5]. From the network perspective, knowledge comes from the process flow which maps information on how an application influences other application's decision making, with different purposes.

The purpose of an application can be decomposed from how an application assesses the situation. Can be described as:

$$(\hat{x} \in [a,b] || req) \therefore \hat{x} \rightarrow Action_i \quad (1)$$

where $\hat{x}$ is an estimate $[a,b]$ data interval, $Action_i$ an action and req is its requirements. Data fusion exploit existing synergies on data to reach better decisions. In a way, fusing information about environmental state with contextual information regarding entities relationship allows data fusion based algorithms to avoid contradictory and redundant actions. When comparing with data fusion, Knowledge fusion considers an additional dimension of errors made by knowledge extractors to decide whether an extracted candidate is consistent with the real world. On IoT scenarios it provides insight to decide whether an application decision is consistent with the real world, also mapping the decision experience from which the relation among applications derive as a common attribute of the linked entities [6]. On knowledge fusion, knowledge is usually represented as a graph [7,8] to represent entities connected by its relations.

In IoT scenarios, a knowledge graph is structured data grouping relationships among applications due to its reasoning process (Eq. (1)). Derived from each execution of its decision making updating a relationship and producing a new instance of the knowledge graph. Newer instances can be stored in a central data repository and used for future analysis allowing knowledge graph refinement.

Reinforcement learning teaches machines by its interactions among the agent, its environment, and the embedded ability to learn from previous experiences [9]. Thus, applying reinforcement learning to map the decision making results allows changing the decision making

* Corresponding author.
E-mail address: gmartinsoc@gmail.com (G. Martins).

https://doi.org/10.1016/j.comcom.2021.04.025
Received 1 December 2019; Received in revised form 25 March 2021; Accepted 22 April 2021
Available online 3 May 2021

hypothesis on the fly. In short, the ability to extract and transfer knowledge is based on how to learn from feedback selectively applying knowledge from previous decision making experience.

We used a multisensor knowledge fusion heuristic (MKFH) on a wireless sensor/actuator network (WSAN) scenario to apply knowledge fusion. Based on MKFH, we introduced two algorithms and proposed architecture as a context model for knowledge, paving the road forward integrated knowledge management on IoT [6]. We do not yet address the challenge of knowledge refinement and validation, indeed essential to create more suitable knowledge graphs. Results are validated by simulation and tests in real nodes.

2. Related work

Knowledge fusion is a way to unveil correlations in a data set [10, 11]. Dong [11] proposed a modelling technique for exploiting correlations among sources and applying it in truth-finding. By using conditional probabilities to express precision and recall metrics, it proposes a function for the occurrence and reliability of a source. The objective of their framework is to distinguish true and false triples in a collection of source outputs. In our approach, we do not assume any knowledge of the inner workings of the sources and how they derive the data provided. In the context of WSAN and IoT, knowledge fusion can be considered as a way to automatically distinguish correct data and erroneous data for creating cleaner sets of integrated data [6].

Indeed, this is the case in practice for many real-world data sources, such as sensors — they provide the data without telling us how it was obtained. Also, even when some information on the data derivation process is available, it may be too complex to reason about; for example, an extractor often learns many patterns (e.g., temperature monitoring may apply differently in a different environment) and uses internal coding to present them; it is difficult to understand all of them when let alone to reason and compare them across sources.

Similarly to [11] our MKFH searches for correlation between sources concerning the rate at which the data is generated and processed. Nevertheless, our heuristic is designed to accomplish that in a resource constraining scenario such as WSAN. Instead of search for provenance, we focus on how an action interacts with the environment when another action is also eligible. In sense, allowing on the fly knowledge transfer and extraction for IoT applications.

Graph-based representation of knowledge allowing map large networks of entities with their relationships their semantic types and properties. Shi [8], presented a framework for triple-context-based knowledge graph that incorporates triple context into the score function, evaluating while learning embeddings instead of using each triple independently. It states that K is a knowledge graph, E and R the set of all entities and relations respectively in K. Each triple is denoted as (h, r, t), in which h is a head entity, t is a tail entity and r is the relation between h and t. That way, all possible relationships regarding the head entity can be mapped with a knowledge graph depending only on its path and neighbourhood context. The neighbourhood context of an entity is the surroundings of it in a knowledge graph and the path context is the set of paths that goes from a head entity to another one in a knowledge graph. Thus, The triple context can be considered to embody the surrounding structures of the graph. In general, the score function of a triple is only related to the embeddings of entities and their relations. By maximizing the joint probability of all triples in the knowledge graph, it defines an objective function that considers information concealed by the graph structures. They adopt a score function and take advantage of negative sampling to approximate it to full softmax functions, having two parts. The first one represents the conditional probability that h is the head entity given the triple context and the second part is the conditional probability that t is the tail entity given the head entity h and triple context. The authors have evaluated this model on link prediction, results have shown significant improvements over the major baselines. However, for real data sets the size of neighbourhood context and path context may be very large, which is computationally expensive.

The creation of manageable knowledge bases be to map contextual and domain-specific information is a way for representing more complex relationships that bind entities in a graph. Likewise [8], this research model the information about how a decision agent evaluates the environment as contextual information, using knowledge graphs embeddings to map previous decision making experiences. However, our integrated approach is designed for resource constrained environments and allow knowledge extraction to map insight on how coexisting application behave together, also transferring knowledge to further integration among different applications. Thus, It enriches data into knowledge to build knowledge graph instances and selectively applies knowledge to enhance the integration of application decisions.

Widely used on the web, knowledge fusion is a way to represent refined knowledge as a graph [7] representing large networks of entities, their relationships, semantic types, and properties. By fusing source, provenance, correlation, or consensual information, it considers errors made by knowledge extractors to automatically decides whether an extracted candidate is consistent with the real world. Dong [11] knowledge fusion method search form correlation between sources concerning the rate at which the data is generated and processed. Shi [8] presented a way to model complex relations among entities on a knowledge graph. Indeed, an important feature for supporting the design of smart services and optimizing the design of smart environments. The large amount of low semantic level data is a vital challenge for allowing refinement of vast knowledge bases and the attainment of knowledge graphs.

3. Knowledge fusion

Knowledge fusion paradigm focus on the integration of information and knowledge from multiple sources to produce less uncertainty, more preciseness, and comprehensible knowledge, as well as revealing hidden and missing knowledge [12–14].Which is expected to be used to enhance decision making, as it may provide better insight and facilitate the situation awareness [10,11,15]. Among the most relevant important applications for knowledge fusion are:

- Intelligent search — Semantic search systems goals are to understand its users and return intelligent answers, in this domain knowledge fusion allows intelligent answers to rise from knowledge.
- Knowledge repositories creation — by providing knowledge that is machine-processable, more complete, less uncertain, and less conflicting, knowledge fusion allows discovering innovative and hidden knowledge needed for the creation and refinement of knowledge repositories.
- Knowledge sharing — is a cycle for knowledge acquisition, conversion, and applicability. In this domain, knowledge fusion allows the recognition and combination of knowledge located and extracted from multiple, distributed, and heterogeneous sources.
- Decision support — regarding semantic content, knowledge fusion is a way to deal with massive amounts of heterogeneous data, information, knowledge compiling it to be used by systems and humans as the basis for decision making.

According to Smirnov [10] Knowledge fusion is widely used to create and refine knowledge repositories. knowledge fusion systems, usually, compute the correctness of a relation among entities based on the agreement between different extractors and priors [7,8]. Dong [15] focus on resolving conflicts from different systems using data fusion inspired technique. By assessing knowledge information regarding how the relationship among two entities are the triples, likewise, [subject, fact, predicate] and associating them with a prior probability allowing the creation or enhancing the completeness of knowledge graphs.

4. Multisensor knowledge fusion heuristic

To set up the environment to execute the MKFH, physical nodes perform hardware initialization, establish communication, initialize all data structures required, and synchronize the WSAN. Its decision making can be divided into the following eight steps:

step 1 — Identify the Decision Environment — the decision making problem is decomposed according to the requirements of its application mapping how it infers the environment state. In this step, the network became aware of each situation are under consideration, and the way to assess them.

step 2 - Acquire the Evidence: encompassing the acquisition of sample data and the estimation of the environmental stare. It consists of reading and adding meaning to the environment, by gathering information regarding its temperature, humidity, luminosity, and so on.

step 3 -Weight the Evidence: processing evidence from each application. The evidence is evaluated from the application's perspective producing a set of alternatives (designed actions).

step 4 - Rate the alternatives: after reaching the set of all suitable actions, it assesses which of them may produce undesirable consequences.

step 5 - Make the Overall Decision: removes from the overall decision every action that may produce undesirable behaviour.

step 6 - Take Action: deals with the deployment of the chosen actions.

step 7 - Revise Overall Decision: after a set of actions takes place the environment may or may not react as expected, in each case, triggering adjustments on the criteria to transfer knowledge.

step 8 - Update the belief system: whenever new criteria emerge from reviewing a decision producing knowledge, it must trigger strategies to transfer this knowledge through the network.

The outcome of step 8 is the way to approach knowledge extraction, this is step generates a history of knowledge graphs instances. Each one represents changes in the cost of choosing a specific action. Thus, those changes when put together map the overall decision making experience, which could be stored in a central repository for future analysis. From this MKFH we introduce two algorithms, Pallas and Ergane.

4.1. System model

Our system model follows one published in [6]. In short, the WSAN is modelled as an undirected graph $G = (V, E)$, where $V = (v_1, v_2, \ldots, v_n)$ represents the set of sensor nodes and $E = (e_1, e_2, \ldots, e_m)$ represents the set of all possible communication links among them. For any given sensor node (SeN), V_i in V, i denotes the index of the SeN that belongs to the WSAN. The SeN is the basic sensing, preprocessing, and actuating unit equipped with at least one physical sensing device or one actuator device. Also, capable of providing one or more tasks depending on capabilities to sample different types of data. All the SeN in the WSAN has a valid communication path to reach the sink node (SkN) which is the gateway between the WSAN and external networks as the Internet. Each SeN stores sensed data in a fusion window data structure (FW) that is an array of readings whose elements have the signal level of abstraction.

For this work, an application is defined by its requisites and the logic from which it assesses a situation (Eq. (1)). Thus, each Application A_i has its requisites mapped as a tuple $(SsR, AA, TP, IR, FR, Cr)$. SsR is a user-defined variable used to characterize the sensing capabilities provided by a given node. SsR denotes the data receiving/sensing rate, meaning the time interval between consecutive readings. AA means the application's action used to interfere in the environment. The TP (Type) element denotes the type of monitored variables, such as temperature, humidity, current, and several others. IR and FR denote the initial (IR) and final (FR) values of the application data range. Finally, Cr is a flag. To summarize, if set to true, the application is critical meaning that its decision making fails whenever it does not reach a decision within a time threshold. Thus, a decision reached on the SeN is final and must bypass the integration procedure.

4.2. Decision context

Context and situation are defined concerning agents and their given purpose. Context is domain specific and defines which situations are under consideration, as situation derives from the perception of the environment state. According to Devlin [16], a feature F is contextual for an action A if the feature F constrains A, and may affect the outcome of A, but is not a constituent of A". Therefore, context premises can be seen as a set of constraints used to reason about a situation. So, contextual information is usually seen as a set of variables or external constraints used to bind entities, attributes, and entity relationships.

As a context-specific information, knowledge can improve situation assessment [6], providing insight and understanding into situation assessment [4,5]. Our contextual model embeds knowledge into autonomous decision making by representing it with a Knowledge Data Structure (KDS):

$$[\pi_i(a|s); \pi_j(a'|s); f(s'|s,a); q(\pi_i(a,s))] \tag{2}$$

where, $\pi_i(a|s)$ is an application i action provoked by the state s; $\pi_j(a'|s)$ is an application j action provoked by the state s; $f(s\}|s,a)$ function for the expected transition from state s under action π_i; $q(\pi_i(a,s))$ value of taking action $\pi_i(a)$ in state s when action $\pi_j(a')$ is also eligible.

4.3. Neighbourhood context

In a multiapplication WSAN, a decision agent assesses environmental readings triggering actions regarding the requisites of the application. A relationship between coexisting applications (r) is constituted by all facts among its actions. A fact derives from the decision making experience mapping how an action ($\pi_i(a)$) of an application (π_i) interfere with other application (π_j) decision making. Considering that an application is an entity in a graph, an application neighbourhood context is its surroundings in the knowledge graph. The knowledge tuple interlinks actions ($\pi_i(a,s); \pi_j(a',s)$) of different entities (applications) through a fact ($f(s'|s,a); q(\pi_i(a,s))$). So, The application context AC represented as a knowledge graph K is:

$$AC_{(\pi_i)} = g(r, \pi_j|\pi_i(a), \pi_j(a')) \forall \{\pi_i, \pi_j, fact\} \in K \tag{3}$$

The graph neighbourhood in Fig. 1 reflect many entity's purpose aspects, such as the goals of π_i, what intersections π_i decision making has with an π_j, which are possible constructive and destructive interactions can exist among them.

4.4. System architecture

Since this study is an extension of Athena algorithm [6], it presents the architecture shown in Fig. 2 to guide implementation of the Athena core Algorithms from our MKFH.

Knowledge consists of facts, rules, representations, and conceptualizations regarding an observed phenomenon [17]. When comparing with data fusion, Knowledge fusion considers an additional dimension of errors which is used to map context. While data fusion plays a vital role in enhancing data in low semantic level into decisions, knowledge fusion deals with the risk of combining these decisions concerning the ownership of the decision in time. To combine data fusion and

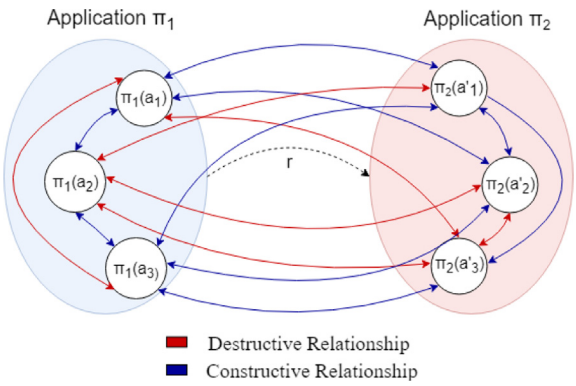

Fig. 1. Knowledge graph example.

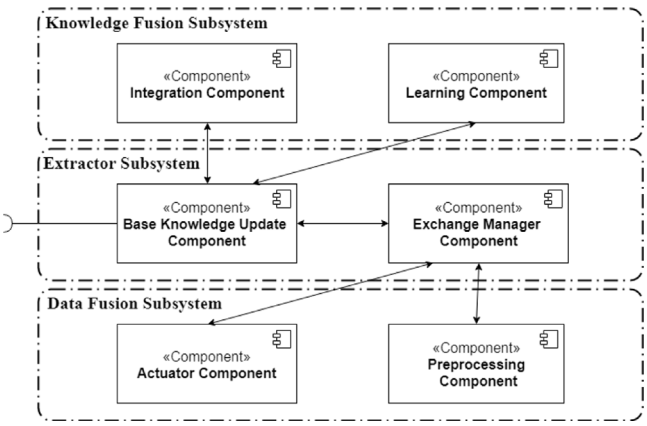

Fig. 2. Logical architecture.

knowledge fusion we need to isolate both processes without losing the decision context. To accomplish that we use the concept of knowledge extractors to link local decisions to its owner as it oversees the overall decision-making process.

Our architecture is designed with a layered architecture pattern with three tiers represented by the following subsystems: (i) data fusion subsystem, (ii) extractor subsystem and (iii) knowledge fusion subsystem.

i Data fusion subsystem is responsible for the decisions deployment. It collects environmental readings and exploits synergies among them, in a way to transform them into data with a higher abstraction level.

ii Extractor subsystem provides connection and interoperability among SkN, actuators and SeNs. It manages the flow of information throughout the decision making process. It is composed of extractors, and function as a source of information for both data fusion and knowledge fusion components.

iii Knowledge fusion subsystem handles an integrated reasoning process and knowledge extraction. In a way, it revises the decision making hypothesis and selectively applies contextual information to enhance the decision experience.

The decision making starts with the application's arrival through the knowledge base update component, which populates knowledge structures, deploys the application's logic, and synchronizes the WSAN. The WSAN gets this information through control messages receipt by the exchange manager component to set up the preprocessing component.

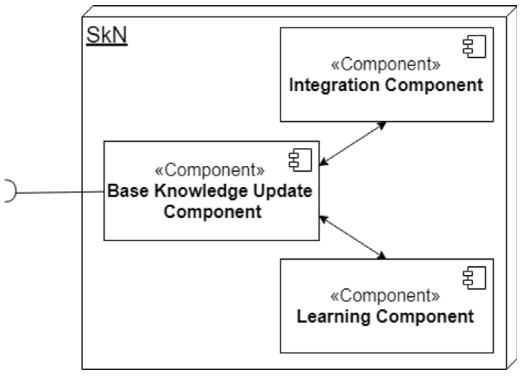

Fig. 3. SkN deployment architecture.

The preprocessing component gathers environmental readings processing them into higher levels of abstraction, enhancing their meaning. This enhanced information is sent to the knowledge base update component with the aid of the exchange manager component. Then, the base update component deals with the arrival of information, in a way to allow extraction of newer knowledge instances on the learning component and integrated decision making on the integration component. The learning component revises the decision making hypothesis and handles its new parameters to be stored by the knowledge base update component. The integration component reaches an overall decision. With aid from the knowledge base update component, the overall decision is sent to the exchange manager component. Which decomposes it into actions and handles it to an actuator component or a preprocessing component, whose must deploy such action.

4.5. Deployment architecture example

Regarding an example for centralized deployment, WSAN consists of a sink node (SkN), several sensor nodes (SeN), and actuators. SkNs are high-end devices acting as exit points from WSAN, SeN nodes contain sensor units and are low-end devices powered by batteries, and actuator nodes are responsible to interfere with the environment according to the application's design. The SkN contains the integration component, learning component, and knowledge base component. The deployment for a SkN follows Fig. 3 diagram. Thus, they are responsible to perceive environmental conditions and to enhance their semantic level of abstraction. SeN contains the exchange manager interface and preprocessing component. The deployment for a SeN follows Fig. 4 diagram. Actuator nodes contain the exchange manager interface and actuator component. The deployment for Actuator nodes follows Fig. 5 diagram.

An important challenge for integrated knowledge discovery and management on IoT is how to extract knowledge regarding its applications decision making [4,18]. A vital constrain for knowledge extraction in IoT systems derive from the vast amount of data in low semantic levels of abstraction gathered by this systems [18,19]. As an MKFH for IoT is an approach towards integrated knowledge management on IoT. It benefits from the Knowledge fusion paradigm to propose a way to exploit synergies from the outcomes of an application decision making, allowing the creation of knowledge repositories from the decision making experience. Thus, this section models the structure behaviour proposed to combine data fusion, learning, and knowledge extraction techniques on IoT scenarios.

Since MKFH derive from an improvement of [6] its algorithms were implemented based on Athena algorithm and we will address them as Athena and Ergane Athena. They also intended to run on a resource-constrained scenario using well-known data fusion, decision-making techniques, and a very simple reinforcement learning approach to pave

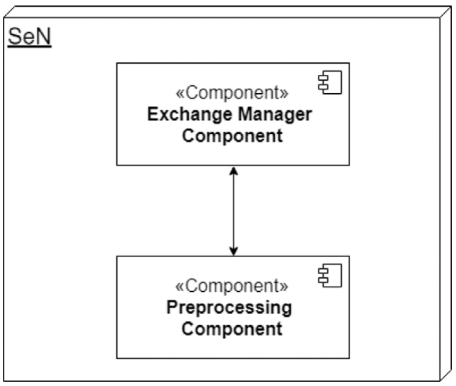

Fig. 4. SeN deployment architecture.

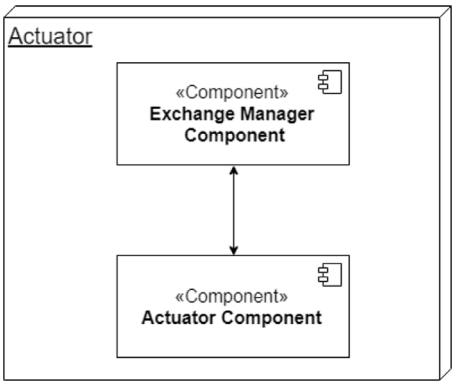

Fig. 5. Actuator deployment architecture.

the road to bolder and complex approaches. Both, Pallas and Ergane decompose the local decision making of their applications into two stages. Stage one occurs on the preprocessing component and focuses on the recognition of events of interest in a resource-constrained environment — to transform data in low semantic level (signal data) into decisions. The major difference between Pallas and Ergane is the way they instance the data fusion model. Ergane completely decouples the logic of its applications from the sensing infrastructure, Pallas SeNs have to be aware of how an application assess the environment. As consequence, while Ergane is a step forward to decentralized decision making, Pallas can take into account if an application is critical allowing it's SeNs to decide and act regardless of any integration. Stage two focus on how the integration among the applications works as the learning agent revise its hypothesis. While Pallas completes the data fusion cycle by reaching a vector of local decisions, Ergane extractor subsystem has to oversee a decision–decision fusion by consulting the application for their decision.

4.6. Athena overview

The execution starts on the knowledge base update component by making the WSAN aware of the active applications — requisites and the logic from which it assesses a situation (Eq. (1)). The same component check for available knowledge data structures to initialize and populate the structures which map the decision context (Section 4.2) and synchronize the WSAN making it operational which ends the Identify the Decision Environment step.

Step 2 (Acquire the Evidence step) start with the execution of the SeN monitoring cycle in the Preprocessing component. In a way, the SeNs collect singular pieces of data (signal level of abstraction), populate a fusion window (pixel level of abstraction), and step 3 (Weight the Evidence) process it into a time-driven summary of the environment condition (feature level of abstraction). On the next step (Rate the alternative), the application logic is consulted to reach the decision level of abstraction by choosing which actions are eligible (Eq. (1)). However, since different applications assess the situation accordingly to their own design, conflicts may emerge producing undesirable results.

To mitigate possible conflicts the Integration component assesses another dimension of the problem and uses the provenance and the decision context to remove actions that are associated with a destructive relationship - a decision whose the risk of deploying them surpasses the risk of not deploying them. That way, the output of step 5 (Make the Overall Decision) is a set of decisions that are suitable and represent minimal risk — the overall decision. Then, the Take Action step deploys the overall decision to the WSAN, under the assumption that it will change the environment behaviour as represented in the Knowledge Data Structure.

Step 7 (Revise Overall Decision) take place at the Learning component to evaluate if the overall decision has changed the environment behaviour as expected — transition described by $f(s'|s,a)$ in the KDS. The Learning component report to the Knowledge Base Update component which adjustments must occur on the existent knowledge structures criteria. After making the suitable adjustments the Knowledge Base Update component sends the new instance of the neighbourhood context to external networks.

4.7. Athena core

Aware of the application requisites, the execution of a data fusion Procedure starts reaching estimates. Then, applications consume the estimates to evaluate the environment reaching decisions and calculating their probabilities. After that, a knowledge fusion procedure revises reached decisions and its hypothesis reaching the overall decision which is sent to physical nodes to deploy its actions. Athena core procedure can be described by the following (Algorithm 1):

Algorithm 1 Athena Main Procedure

Require: Knowledge Set KDS= (k1,..,kn);
New application Set NAS = (A1,..,An).
Ensure: Overall decision set ODS= (od1,..,odn).
1: **procedure** PALLAS(KDS,NAS).
2: **if** NAS is empty **then**
3: Apply Preprocessing Procedure
4: Exchange Manager Procedure
5: Apply Learning Procedure
6: Apply knowledge base Update Procedure
7: Apply Integration Procedure
8: Send ODS to SeN
9: **end if**
10: **if** NAS is not empty **then**
11: Apply knowledge base Update Procedure
12: **end if**
13: **end procedure**

4.8. Pallas athena

Pallas SeN executes the Moving Average Filter (MAF) algorithm found on [2] to reach estimates. Then, applications consume the estimates evaluating them regarding their requisites, in a way reaching decisions and calculating its probabilities. At the SkN the learning step evaluates the previous execution modifying the costs value through a reinforcement learning approach. A change on a knowledge tuple element triggers the knowledge base update procedure. After that, the integration procedure processes a set of Bayesian hypothesis tests to decide which decisions must compose the overall decision. Which are sent to the physical nodes capable of deploying these actions.

4.9. Pallas data fusion subsystem

Pallas data fusion subsystem occurs only on SeNs and is responsible for acquiring environmental information, assess its situation, and deploy actions accordingly. The preprocessing component implement Steps 2 (Acquire the Evidence) and 3 (Weight the evidence) and the Actuator component implements Step 6 (Take Action). The preprocessing procedure (algorithm 2) periodically triggers the sensor unit collecting (line 1) a sample of the environmental condition (signal-level information). These environmental readings are stored in a Fusion Window data structure. The pixel level is achieved by populating FW. In possession of a pixel (a vector of samples), an estimate of the environment state is reached, which has a feature level of abstraction. Then, Pallas consult its applications that evaluate which action is required, reaching a decision level of abstraction (line 5). Then, the Cr flag is checked, if Cr is true the action is instantly deployed. After reaching a decision the SeN calculates a probability for generating the decision. In a way, each value of $\hat{x}$ is an observed value of random variable X, meaning the sensed data has triggered action. So there is a probability function that describes the probability of the MAF outcome trigger that action [20]. Finally, a symbol (also decision level of abstraction) composed by the MAF outcome and a probability set is compiled into a message in a way forming an occupancy grid feature map described at [2]. The SeN and sent (line 6) to the SkN. Then the SeN enters the power-saving state (line 7) until the next sampling time to decrease the energy consumption. An Actuator procedure is responsible to deploy each decision by triggering actuator units to interfere with the environment accordingly.

Algorithm 2 Pallas Preprocessing Procedure

Require: Application set A = (a1,..,an).
Ensure: Decision set D;
1: **procedure** PREPROCESSING(A)
2: Collect ESD (environment sample data) ▷ signal level
3: populate FW ▷ pixel level
4: Apply MAF ▷ infer the environment feature
5: **for** each application **do**
6: Consume MAF output ▷ decision Level
7: **if** Cr is True **then**
8: Append D to ODS
9: send ODS to exchange manager procedure
10: **end if**
11: Calculate probabilities
12: **end for**
13: Compile D
14: Send D to SKN Procedure
15: Sleep for SsR time
16: **end procedure**

4.10. Pallas extractor subsystem

Pallas extractor subsystem is responsible to deal with the exchange of information among SkN, SeNs, and external networks. The Knowledge Base Update Component implement Steps 1 (Identify Decision Environment) and 8 (Update Belief System) and the Exchange Manager component implements Steps 1 (Identify Decision Environment). The knowledge base update procedure described in algorithm 3 runs on the SkN and is triggered by (i) application arrival, (ii) new version of the knowledge base availability, (i) the receipt of a symbol from SeNs, and (iv) new knowledge produced on learning procedure.

Concerning the case (i) when an application arrives in the network (line 2), this procedure receives the knowledge tuple that complete the knowledge base (line 4), the applications requisites, and its rules. After that, Pallas checks the received knowledge tuple (KDS). When a knowledge tuple is invalid. Pallas algorithm set a standard tuple in which the cost (C) is set equal to one and the ES is set to "equal". Whenever all knowledge tuple is considered valid, the knowledge base update procedure deploys the application's logic into the network (line 7). This procedure also synchronizes the WSAN (line 8).

In case (ii) a new version of the knowledge base is available (line 10), This procedure receives the new version from outside the network and updates the knowledge base with the newer knowledge tuple (line 11). Case (iii) occurs from symbol receipt from SeN, which triggers the retrieval of information need by the integration procedure and learning procedures In case (iv), the Knowledge Base Update Procedure updates the value of a cost at the knowledge tuple (line 11) to meet the output of the Learning Procedure (lines 10 and 18).

Algorithm 3 Pallas Knowledge Base Update Procedure

Require: Overall Decision set ODS,
Communication links E = (e1 ... , em),
New Application set NAS = (na1,..,nan),
New Knowledge set NK = (nk1,..,nkn),
New Cost set NC= (nc1,..,ncn),
Decision set D.
Ensure: Knowledge set KDS =(k1,..,kn);
1: **procedure** KB UPDATE(NAS,NK,NC,D,E,ODS)
2: **if** NAS is not empty **then** ▷ deal with new application
3: **for** each element in NAS **do**
4: Update KB ▷ Knowledge base completeness
5: Prepare A
6: **end for**
7: Deploy A on WSAN ▷ deploy application's logic
8: Synchronizes WSAN
9: **end if**
10: **if** Nk is not empty **then** ▷ new knowledge available
11: Update KB ▷ updates the Knowledge base
12: **end if**
13: **if** D is not empty **then**
14: **for** each element in D **do**
15: get KDS ▷ get relationship among actions
16: **end for**
17: Call Learning Procedure
18: Call Integration Procedure
19: **end if**
20: **if** ODS is not empty **then**
21: Send ODS to SeN
22: **end if**
23: **end procedure**

The Exchange Manager procedure (algorithm 4) occurs on the SeN set the sample periodicity to meet application's requirement and handles all information exchange among SkN and SeNs.

4.11. Pallas knowledge fusion subsystem

Pallas Knowledge Fusion subsystem occurs on SkN and is responsible to reach the overall decision and revise de decision hypothesis. Pallas learning component implements Step 7 (Revise Overall Decision) and the integration component implements steps 4 (Rate the Alternatives) and 5 (Make The Overall Decision). Pallas learning procedure (algorithm 5) adapt reinforcement learning to change the hypotheses cost on the fly. It portrays the success or failure of the last overall decision by comparing the expected state with the current decision set (line 3). If the inferred state is equal to the expected state, the overall decision was successful and the costs involved in it must be lowered. Otherwise, the overall decision was a failure, so the costs must be raised. That way, a reward (Eq. (11)) mapping positive or negative feedback is applied to change the cost value. So, the Learning procedure

Algorithm 4 Pallas Exchange Manager Procedure

Require: Application set A = (a1,..,an)
Communication links E = (e1 ... , em),
Overall Decision set ODS.
Ensure: Decision set D.
1: **procedure** EXCHANGE MANAGER(A,E,ODS)
2: **if** A is not empty **then**
3: set monitoring requisites
4: Call Preprocessing Procedure
5: **end if**
6: **if** A is empty **then**
7: send D to SkN
8: **end if**
9: **if** ODS is not empty **then**
10: send ODS to actuators
11: **end if**
12: **end procedure**

result is a calibration for a decision expected loss, an update of the beliefs involved in the decision making as a consequence of revising the way a decision was made.

A cost maps an action relationship. This way, higher costs indicate higher expected losses resulting from interference. Thus, bigger costs must lead to smaller changes resulting from an experience. For this reason, the cost increment is inversely proportional to the prior knowledge about the relationship. However, this characteristic makes it possible to output extremely large increments whenever the cost is small. They, in turn, may lead to the undesired behaviour of the algorithm. To solve this problem a specialist may set a proper upper limit for the increment value (SL).

Algorithm 5 Pallas Learning Procedure

Require: Decision set D = (d1,..,dn),
Knowledge set KDS =(k1,..,kn),
Last MAF output = last_E,
Current MAF output = current_E.
Ensure: New Cost set NC= (nc1,..,ncn),
Decision set D = (d1,..,dn).
1: **procedure** LEARNING(D,KDS,last_E,current_E)
2: **for** each element in D **do**
3: distance = current_E - last_E
4: Compare distance with ES
5: Apply Reinforcement
6: **end for**
7: last_E = current_E
8: Call Knowledge Base Update Procedure
9: **end procedure**

In the integration procedure, the overall decision is reached by choosing the set of actions that lead to a smaller risk for the network. In a way, the decision set with lower loss is achievable by mitigating the contradiction or redundancy within the elements of this decision set. Let us Consider a set up composed of two applications $A = \pi_i, \pi_j$. Whenever an environmental reading belongs to an interest region for π_i and π_j, Pallas decision making process unfold into tree possible outcomes: both decisions are right (i), one decision is right and the other is wrong (ii) and both decisions are similar (iii). Regarding case (i) both decisions must take place, will be a cost resulting from not applying one of them. In case (ii) the right decision must be deployed and the wrong discarded. Finally, in case (iii) only one of them must be deployed, will be a cost resulting from applying them together.

Thus, Pallas assumes that its applications are trying to decide which actions are a better fit a monitored environmental behaviour. At algorithm 2, Each application has compared the observed environmental

Algorithm 6 Pallas Integration Procedure

Require: Decision set D = (d1,..,dn),
Knowledge set KDS =(k1,..,kn),
Current MAF mean = current_E,
Probabilities set P = (p1,..,pn).
Ensure: Overall decision set ODS.
1: **procedure** INTEGRATION(D,KDS,current_E,P)
2: **for** each element in D **do**
3: Calculate risk scores
4: **if** RiskH0 <= RiskH1 **then**
5: Add decisions in ODS
6: **end if**
7: **end for**
8: Prepare ODS
9: Call Knowledge Base Update Procedure
10: **end procedure**

Algorithm 7 Ergane Preprocessing

Require: Sensing rate SsR.
Ensure: Feature map FM;
1: **procedure** PREPROCESSING(SsR)
2: Collect ESD (environment sample data) ▷ signal level of abstraction
3: populate FW ▷ pixel level
4: Apply MAF ▷ feature level
5: Compile FM ▷ decision level
6: Send FM to exchange manager procedure
7: Sleep for SsR time
8: **end procedure**

reading with its requisites and should opt to interfere or not interfere with the environmental conditions. At algorithm 5, Pallas adjusts the integration hypothesis, by changing the costs involved in previous integration. This revision modifies the context in which the overall decision was made, as well as maps it with a new instance of the knowledge graph. At algorithm 6, Pallas assess the risk of every application decision, as well as compiles the least risky one into an overall decision. This step mitigates contradictions and redundancies, as such actions represent losses over the decision experience. However, to reach decentralized decisions allowing to deal with critical applications, Pallas preprocessing step is not fully decoupled from its knowledge fusion step. Addressing this feature we introduce another algorithm, called Ergane.

4.12. Ergane athena

Different from Pallas, Ergane approach completely decouples the WSAN from its applications. So, SeN does not gain knowledge about the requisites of Ergane's applications, acting only as a collector node. Instead, the SkN configures a sample time period meeting the most demanding application. Periodically, by executing the data fusion procedure, Ergane's SeNs reach symbols. Which are compiled into a message and sent to the SkN (line 2). This approach does not deal with critical applications, so all applications must became integrated before reaching a decision. After configuring a sample periodicity and the WSAN is synchronized, Ergane preprocessing procedure starts. Its execution occurs only at the SeN and implements a data fusion algorithm. At the SkN, first, the learning procedure (line 3) uses the information gathered from the last execution of the decision making to generate a new instance of the knowledge graph through a reinforcement learning approach (line 4). After that, Acting as a decision node the SkN prepare the symbols to be consumed by the applications reaching decisions. The integration procedure calculates a probability for the symbol to trigger

each decision. Then, apply the Bayesian hypothesis test to integrate the application's decisions forming the overall decision (line 5). Finally, integrated decisions are sent to physical nodes capable of acting, in turn, deploying the overall decision (line 6).

4.13. Ergane's data fusion subsystem

Ergane's data fusion subsystem occurs only on SeNs and is composed of a preprocessing component that implements Step 2 (Acquire the Evidence) and an actuator component that implements Step 6 (Take Action). Ergane preprocessing procedure found on Algorithm 7 occurs on SeN periodically triggering the sensor unit to produce a sample of the environmental condition (signal-level information). The pixel level is achieved by populating FW. In possession of pixel data (a vector of samples), Athena applies a Data Fusion method (line 2), which output has a feature level of abstraction and is a mean of FW. Finally, this output is compiled into an occupancy grid feature map [2] containing estimates for the monitored physical phenomena. That way, Ergane reaching a symbol with a decision level of abstraction. An Actuator procedure is responsible to deploy each decision by triggering actuator units to interfere with the environment accordingly.

Algorithm 8 Ergane's Knowledge Base Update Procedure

```
Require: Overall Decision set ODS,
         Communication links E = (e1 ... , em),
         Application set A = (a1,..,an),
         New Knowledge set NK = (nk1,..,nkn),
         New Cost set NC= (nc1,..,ncn),
         Feature map FM.
Ensure: Knowledge set KDS =(k1,..,kn);
         Decision Set D,
         Probability Set P.
 1: procedure KB UPDATE(A,NK,NC,FM,E,ODS)
 2:     if NAS is not empty then  ▷ deal with new application arrival
 3:         for each element in NAS do
 4:             Update KB      ▷ ensures Knowledge base completeness
 5:             Prepare A
 6:         end for
 7:         Deploy A on WSAN    ▷ deploy application's logic on SeNs
 8:         Synchronizes WSAN
 9:     end if
10:     if Nk is not empty then              ▷ new knowledge available
11:         Update KB                        ▷ updates the Knowledge base
12:     end if
13:     if FM is not empty then
14:         for each element in A do
15:             for each element in FM do
16:                 Append decision to D
17:                 Calculate Probabilities
18:                 get KDS                  ▷ get relationship among actions
19:             end for
20:         end for
21:         Call Learning Procedure
22:         Call Integration Procedure
23:     end if
24:     if ODS is not empty then
25:         send ODS to SeN
26:     end if
27: end procedure
```

4.14. Ergane's extractor subsystem

Ergane extractor subsystem is responsible to deal with the exchange of information among SkN, SeNs, and external networks. The Knowledge Base Update Component implement Steps 1 (Identify Decision Environment) and 8 (Update Belief System) and the Exchange Manager component implements Steps 1 (Identify Decision Environment). In a way, the Knowledge Base Update procedure starts when (i) an application arrives in the network, (ii) a new version of the knowledge base is available, (iii) a new knowledge graph instance is extracted, and (iv) an application reach a decision. The case (i) populate knowledge structures add completeness for the knowledge graph (line 4) if a KDS is invalid Athena algorithm set a standard tuple in which the cost (C) is set equal to one and the ES is set to "equal". After that, the deployment of the application's most demanding sensing rate parameter occurs through a set of control messages (line 7). Then, the WSAN is synchronized (line 8). In case (ii) the knowledge graph is fully updated (line 11), upon the receipt of a new version from outside the network (line 10). Case (iii) occurs as a consequence of the leaning procedure (line 10). The value of a cost C is updated (line 11). Case (iv) occurs as preparation for the integration step and is responsible to retrieve the KDS tuples related to the application's decisions (line 18).

Ergane's Exchange Manager procedure implements (algorithm 9) occur on the SeN setting sample periodicity to meet application's requirement and handling all information exchange among SkN and SeNs.

Algorithm 9 Ergane Exchange Manager Procedure

```
Require: Overall Decision Set ODS,
         Sampling rate SsR,
         Communication links E = (e1 ... , em);
Ensure: Feature map FM,
         Overall Decision Set ODS.
 1: procedure EXCHANGE MANAGER(SsR,E,ODS)
 2:     if SsR changes then
 3:         set monitoring requisites
 4:         Call Preprocessing Procedure
 5:     end if
 6:     if SsR does not change then
 7:         Call Preprocessing Procedure
 8:     end if
 9:     if FM is not empty then
10:         Send FM to SkN
11:     end if
12:     if ODS is not empty then
13:         send ODS to actuators
14:     end if
15: end procedure
```

4.15. Knowledge fusion subsystem

Ergane's knowledge fusion Subsystem occurs on the SKN implementing Steps 4 (Rate Alternatives), 5 (Make Overall Decision), and 7 (Revise Overall Decision) from the MKFH. Ergane learning procedure applies the reinforcement learning approach described in algorithm 10.

The integration procedure (algorithm 11)receive a feature map as input, then it consult the applications reaching decisions and calculating probabilities for each decision. These decisions are combined with knowledge to produce the overall decision mitigating redundant nor contradictory actions.

Likewise in Pallas, let us Consider a setup composed of two applications. When an environmental reading belongs to an interest region an application will add an action to a decision set D. Ergane's overall decision aims to be least costly and most beneficial as possible. Therefore, whenever a reading triggers two applications to add an action in D. Both actions are right and must be deployed. One Action is right and the other is wrong or both decisions are similar, in each case, only one of them must be deployed. To reach an overall decision, Ergane calculates the probability of a decision is right and performs a hypothesis test for each Action in D.

Algorithm 10 Ergane Learning Procedure

Require: Feature map FM,
Knowledge set KDS =(k1,..,kn),
Last MAF output = last_E,
Current MAF output = current_E.
Ensure: New Cost set NC= (nc1,..,ncn),
Feature map FM.
1: **procedure** LEARNING(D,KDS,last_E,current_E)
2: **for** each element in FM **do**
3: distance = current_E - last_E
4: Compare distance with ES
5: Apply Reinforcement
6: **end for**
7: last_E = current_E
8: Call Knowledge Base Update Procedure
9: **end procedure**

Algorithm 11 Ergane Integration Procedure

Require: Decision set D,
Probability set P,
Knowledge set KDS =(k1,..,kn).
Ensure: Overall decision set ODS.
1: **procedure** INTEGRATION(FM,P,KDS)
2: **for** each element in FM **do**
3: Reach application's decisions
4: Calculate risk scores
5: **if** RiskH0 <= RiskH1 **then**
6: Add decisions in ODS
7: **end if**
8: **end for**
9: Prepare ODS
10: Call Knowledge Base Update Procedure
11: **end procedure**

5. Use case implementation

This topic describes the elements applied to develop a prototype to evaluate our MKFH when applied on a hypothetical Smart Grid scenario found in [21–23].

5.1. Motivating scenario

From the IoT point of view, the Smart Grid concept aim increasing efficiency in the delivery of electric power [24,25]. According to Momoh [24] Smart Grids are a self-healing network which uses real-time measurements to increase reliability and to improve assets management for an electric grid. By allowing on the fly monitoring, the employment of WSANs on Transmission towers support integrated management of the electric power distribution networks concerning the Smart Grid physical integrity and supply–demand balance [25].

In [22,23], a smart grid scenario was represented with two applications sharing communication and sensing infrastructure as described in Section 5.2. According to [26], increasing loads of the power lines had become an issue of great importance, indeed, as the load increases more electricity turns into heat making high temperature values evidence of energy waste and potential operational failures. Thus, knowledge about the power line temperature is an important feature for making decisions about how to manage the power line's load. Also, batteries are used as a solution to unexpected peaks in the electricity demand. However, batteries may also overheat and cause structural damage to towers.

Our proposed scenario, transmission towers are electric power transmission towers endowed with a redundant energy storage system (local batteries) and sensors understood as a computational agent capable of autonomous decision making and action. Also, the temperature behaviour on the power line and the battery can affect the monitored temperature of each other. On transmission towers with manageable redundant battery system, on temperatures at 75 °C the power line breaks [26], the safe line temperature is around 40 °C–65 °C and the usual battery temperature goes from 40 °C–144 °C [26].

5.2. Applications description

Aligned with this hypothetical scenario, a smart grid composed of 4 power lines and a redundant battery system was considered. The OPLM application goal is to preserve the power line health and optimize the amount of energy transmitted. The presence of overheating in the overhead power lines could endanger them. Batteries are seen as a solution to mitigate transmission failures, solving unexpected peaks in the electricity demand and avoid damaging the battery due to overheating. Regarding thermal models in [27], the applications decision making problem could be decomposed into 4 situations as follows:

- Situation 1 (S1) maps a condition in which the power line and the battery condition is considered normal, in a way the temperature does not indicate imminent risk. For this situation temperatures, readings are expected to vary from $40°C$ to $65°C$. To conclude, the OPLM and battery application should not interfere with the environment.
- Situation 2 (S2) maps the condition in which only the power line is operating on a critical state, but temperature readings do not indicate imminent risk for the battery. For this situation temperatures, readings vary from $65°C$ to $75°C$, implying that OPLM application must reduce the amount of energy transmitted on the power line.
- Situation 3 (S3) maps a condition which the power line is operating normally and the battery is operating on a critical state. That way, the temperature is being affected by the battery condition. At this situation temperatures, readings vary from $75°C$ to $144°C$. Thus, only the battery application should act, switching its energy storage system.
- Situation 4 (S4) maps a condition in which the temperature indicates the power line and battery failure. For this situation temperatures, readings are expected to vary from $144°C$ to $300°C$. Implying failure, so the energy load should be redirected and both systems shutdown.

For situation 3 the OPLM application decision standalone is contradictory, by implying imminent damage on the power line when the temperature change is a function of a battery overload. Situation 4 present behaviour in which both applications could decide properly, however a combined decision from both application waste energy. That way, this implementation considers that only the OPLM decision application must apply for situation 4.

5.3. Environment description

The experiment was conducted on real and virtual Zolertia Z1 platform. Z1 nodes are endowed with 8 kB of RAM, 128 kB of flash memory for program storage, 1 Mb for data storage and are powered by a 800 mAh Li-ion battery. Our Algorithm was built by using Contiki OS 2.7 [28]. Regarding the WSAN, is composed of 8 SeNs, 1 actuator node and 1 SkN. Its SeN can play the role of either a fusion node or a collector node, where the collector nodes are responsible for collecting data and the fusion nodes are responsible for applying the MKF algorithm and to manage all SeNs within the system. In this theoretical scenario, each power line has two sensor nodes, embed two temperature sensor units [21]. In a way, each collector node gathers four temperature readings, all simulations consider that at least one reading is wrong. Each sensor node has an implementation of the rime communication stack.

All simulations run on an Instant Contiki 2.7, a virtual environment with a single-core processor and 1 GB RAM and memory used Cooja simulator, an accurate and scalable simulator for WSAN [29,30]. Simulations considered three experiments setups. First, setup E1 assessed the overhead in terms of resource consumption, then E2 and E3 assessed our MKFH in terms of its decision making accuracy. Setup E2 set a baseline for comparing our algorithms with MAF [2,23] and setup E3 assessed the accuracy as a function of a value function. Each experimental setup executed simulations regarding the conditions of the situations described in Section 5.2. For each situation, simulations sustained at least 1 h and repeated to reach a confidence interval of 95% for its results.

5.4. Experiment constrains

The development of the experiments unveiled two important constrains. First, the Smart Grid scenario deals with high temperatures. Thus, it was unfeasible to reproduce the physical conditions described at Section 5.2 on a laboratory environment. A way to overcome this limitation was the adoption of a simulation methodology which gradually input temperature readings for each sensor unit. The second limitation derives from the fact that by generating data for the described scenario, is unfeasible to determine when error derives from the environment model, from environmental changes or due hardware malfunction. Also, for the situations described in Section 5.2, erroneous decisions occur more often as the environment temperature reaches the application's critical region boundaries.

Regarding the second limitation, experiments used thermal models in [27]. Thus, the initial temperature was set at 40 $°C$, the interval between samples were set to 15 s. As the simulation advances, the temperature increases and a random error limited is added by the sensor unit precision. Also, to map possible transitions, at least one reading of the fusion window is generated randomly outside the range of the considered situation. This approaches may not always represent real applications, however, the diversity provided is sufficient for this group of experiments as explained in [23,31,32].

5.5. Metrics

To evaluate the overhead in terms of resource consumption this work has focused on memory and energy consumption. Memory consumption is defined as the amount of memory used by our algorithm when installed in the nodes (RAM and ROM). Analogously, the energy consumption is a measurement for the amount of energy a sensor node consumes on a time period. The metrics used in the resource consumption experiments were:

- Used bytes in % RAM memory — the proportion of the total amount of RAM memory in the Zolertia Z1 platform.
- Used bytes in % program memory — the rate among the amount of memory used to deploy our algorithms software components on Zolertia Z1 platform and the total flash memory available.
- Average node energy consumption for decision cycle — the total energy spent by a Zolertia Z1 node during a single execution of the decision making.

To evaluate how often decisions correctly represent the experimented situation, his study has used definition of trueness, precision and accuracy presented in [33,34].

- Trueness: is the closeness of agreement between the average value obtained from a large series of test results and an accepted reference value. Thus, It expresses the matching between a measurement and an accepted reference quantity value. It derives from how much a measurement is distant from the reference. For example, as much as a sample mean estimate converges to the population mean it may be considered the real value for the population mean without being equal to it. The difference between a single estimate and the real value express its trueness.
- Precision: is the closeness of agreement between independent test results obtained under stipulated conditions. Thus, precision depends only on the distribution of random errors and does not relate to the true value, is usually expressed in terms of imprecision and computed as a standard deviation. Less precision is reflected by a larger standard deviation.
- Accuracy: The closeness of agreement between a test result and the accepted reference value. Thus, it derives from a combination of random components and a common systematic error or bias component. That way, accuracy derives explicitly from trueness and precision, hence, accuracy would be linked to a quantity related to the total measurement error (both systematic and random error).

According to [33,34], accuracy refers to the combined result regarding a systematic component of the measurement error (related to the trueness) and a random one (related to the precision). Trueness refers to the proximity of a result to its true value, an estimate of a systematic measurement error or a bias. Precision refers to the repeatability of a result, describes the spread of results obtained under a specific measurement protocol. Therefore, the MKFH decision making is accurate when its decisions meet the desired behaviour for the experimented situation, a suitable integrated decision. To assess accuracy, our algorithms were subjected to a situation reaching decisions. Whenever a redundant or contradictory decision was detected or classified correctly as a healthy state, it was accounted as an accurate behaviour. Analogously, incorrect detection of a situation or change on the environment is missed was accounted as inaccurate behaviour. Then, for each simulation execution, a measurement for the precise behaviour proportion was accounted and an interval of plausible values for the proportion of suitable integration was estimated following Eq. (4). As stated in [33], the precision on independent test results obtained under stipulated conditions depends only on sample standard deviation ($\sigma/n^{-1/2}$). Thus, as the number of repetitions the sample standard deviation converges to the true value [20]. For each simulation, we calculated a confidence interval until reaching a confidence interval of 95%.

$$\overline{X} \pm T_{n-1}^{-1}(c)\sigma/n^{-1/2} \tag{4}$$

where $\overline{X}$ is precise behaviour proportion mean; $T_{n-1}^{-1}(c)$ is a T-student statistic; σ the standard deviation and; n the number of repetitions.

5.6. Data fusion method

We used the Moving Average Filter algorithm (MAF) found in [2] as data fusion technique which, as the name suggests, is filter computes the arithmetic mean of a number of input measurements given an input digital signal $\mathbf{z} = (z(1), z(2), \ldots)$. The goal is to estimate the true value of the environmental parameter $\hat{\mathbf{x}} = (\hat{x}(1), \hat{x}(2), \ldots)$ that is estimated by:

$$\hat{x}(k) = 1/M \sum_{i=0}^{M-1} z(k-1) \tag{5}$$

For every $k \leq M$, where M is the number of input observations to be fused and k the current input measurement. The sensor nodes estimate events based on simple Moving Average filters that is used to improve the sensor readings. For evaluating our algorithms, MAF implementation used a fusion window that stored 5 values. The collector nodes were endowed with two sensor units, each of which were trigger to collect two redundant environmental reading. Considering T_i current execution, the fusion window is populated as Fig. 6.

The use of the mean as an estimator may reduce errors of tracking application's interest region. On a multi-application WSAN, to reach decisions an application reasoning process may be implemented as conditional rules (Eq. (1). Coexisting applications may assess the same environmental reading differently, which may result in conflicting state

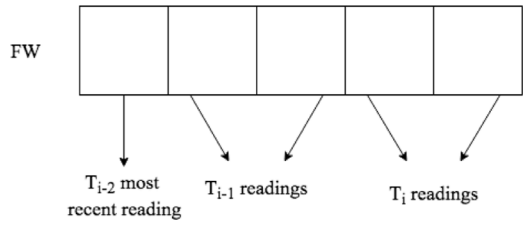

Fig. 6. MAF's fusion window example.

estimates $(Action_i, Action_j)$ for the environment condition described by following equations:

$$(\hat{x} \in [a,b] | req_{\pi_i}, \pi i). \therefore \hat{x} \rightarrow Action_i \tag{6}$$

$$(\hat{x} \in [a,b] | req_{\pi_j}, \pi_j). \therefore \hat{x} \rightarrow Action_j \tag{7}$$

Though the use of data fusion, MAF sends only higher abstraction level data to the applications, as consequence it reduces the length of sent messages which leads to a less energy consumption [21,35].

5.7. Probability estimates

Regarding the probability calculation, let' consider the vector $(\hat{Z}_1, \ldots, \hat{Z}_n)$ form a feature map. each $\hat{Z}$ is an estimate for the environmental parameter obtained as a mean of FW, an environmental readings sample. As a premise, the probability for an application's decision is right depends only on the trueness of its situation recognition. So, the probability for an application to be right is the same that model the probability distribution of the errors of $\hat{x}$ regarding a population parameter.[20,36], under the assumption that an estimate error $(\theta_i - \hat{\theta}_i)$ is a normally distributed around the parameter true value. An estimator $\hat{\theta}$ is also normally distributed. However, values of μ (population mean) and σ (population standard deviation) are unknown and must be estimated from the application's requisite. Thus, the estimator $\hat{\mu}$ and $\hat{\sigma}$ must be a function of the minimum and the maximum value of the applicable interest region $I = [a,b]$. This work uses a simple way to estimate population mean and standard deviation.

$$\hat{\mu} = (a+b)/2 \tag{8}$$

$$\hat{\sigma} = b - \hat{x}/3 \tag{9}$$

$$P(X = \hat{x}) = \Phi(z \leq (x-\mu)/\sigma)n \tag{10}$$

5.8. Knowledge fusion method

The MKFH deal with knowledge transfer and extraction through the decision experience of an integration agent. The main goal is to discover and embed knowledge from previous decision making experience. Building the understanding about how coexisting application correlates from interaction with the environment and applying it to enhance integration.

Bayesian test procedure provides a simple and consistent way to reason in presence of uncertainty. To decide how to integrate conflicting actions we use a Bayesian hypothesis test procedure. To accomplish knowledge extraction, a reinforcement learning agent receive rewards mapping the environment feedback into a value function C_{ij} (Eq. (11)) which is also context variable used to map AC (Section 4.3). To transfer this knowledge, our heuristic uses the value function as problem variable to perform integration, in a way the overall decision making described at Section 5.10 derive from the value function stored on the knowledge base (Section 4.2) and π_i probability.

5.9. Reinforcement learning approach

When a puppy learning a new trick it has no one to make it easier to understand what it is asked for, the only information source is how the world around it reacts to its actions. By rewarding the puppy's actions to reflect which one was right or wrong, its owner may exploit a set of interactions to produces a wealth of information about cause and effect. That way, the puppy builds knowledge about what to do to achieve goals. A similar approach is suitable to teach an agent who seeks to achieve a goal about tradeoffs involved in making a decision.

The agent responsible for the overall decision derive its value function from the costs involved in each possible combination of applications decisions. The reward is a function of environment feedback, in a way positive feedback leads to a positive reward (decreasing costs) and negative feedback to a negative one (increasing costs). The basic idea is to reward interactions with the environment accordingly to its feedback. The learning agent may capture the most relevant aspects of the problem, also discovering which actions yield the most reward. So, the Learning procedure result is a calibration for a decision expected loss, as consequence, this procedure adjusts the understanding of the relationship among different applications. The reward function follow Eq. (11):

$$reward = \begin{cases} -Cost^{-1} & \text{, positive feedback.} \\ +Cost^{-1} & \text{, negative feedback.} \end{cases} \tag{11}$$

5.10. Integration approach

We consider a set up composed of two applications $App = app1, app2$. Whenever an environmental reading belongs to an interest region for $app1$ and $app2$ (Eqs. (6) and (7)), the decision making process unfold into three possible outcomes: both decisions are right (i), one decision is right and the other is wrong (ii) and both decisions are similar (iii). Regarding case (i) both decisions must take place, maybe a cost resulting in not applying one of them. In case (ii) the right decision must be deployed and the wrong discarded. Finally, in case (iii) only one of them must be deployed. For case (ii) and (iii) there are losses on applying these decisions together, so a cost is produced from applying both of them.

Therefore, an consistent way to produce an overall decision is to choose which parameter space Ω_0, Ω_1 is the most probable origin for the evidence with regard of the relative worth of a gain associated with the decision, therefore, which hypothesis H_0, H_1 minimizes the overall decision risk. By choosing to reject H_1 (alternative hypothesis) imply that Ω_0 is the parameter space which is the most probable origin for the evidence. So, the decision π_i minimizes the overall decision risk and π_i must become part of the overall decision. Analogously, rejecting H_0 (null hypothesis) means that the decision π_i represent unnecessary risk.

As defined in [20] a function that assigns to each possible amount $x \in (-\infty < x < \infty)$, a number $U_{(x)}$ representing the actual worth produced by a gaining x is called an utility function. It is designed to represent the relative worth of a gain obtained by considering that an action π_i is right a priori.

$$U_{\pi_i} = U_x / \int_D U_{(x)} dx \tag{12}$$

The hypothesis test procedure follow the steps below:

- Calculate $Risk(H0)$ and $Risk(H1)$:

$$\begin{bmatrix} Risk(H0)/U_{\pi_i} \\ Risk(H1)/(1-U_{\pi_i}) \end{bmatrix} = \begin{bmatrix} P(X=\hat{x}) \\ 1-P(X=\hat{x}) \end{bmatrix} \begin{bmatrix} \sum_{j=1}^{n} C_{ji} \\ \sum_{j=1}^{n} C_{ij} \end{bmatrix}^T \tag{13}$$

$P(X = \hat{x})$ is the probability of the sensed data trigger π_i,
C_{ij} is the cost of applying π_i given π_i^c,
$\sum_{j=1}^{n} C_{ji}$ is the cost of applying π_i^c given π_i and U_{π_i} is the utility of taking π_i.

Table 1
Pallas in terms of memory consumption.

Node/memory	RAM	Program memory
Collector node	6.9%	2.5%
Fusion node	5.4%	6.7%

Table 2
Pallas in terms of energy consumption.

MKF	Pallas Collector node	MAF Collector node
S1	(4.980 ± 0.538) µJ	(4.540 ± 0.244) µJ
S2	(4.560 ± 0.260) µJ	(4.516 ± 0.1800 µJ
S3	(4.833 ± 0.233) µJ	(4.640 ± 0.0970 µJ
S4	(4.724 ± 0.336) µJ	(4.680 ± 0.1270 µJ

Table 3
Ergane in terms of memory consumption.

Node/memory	RAM	Program memory
Collector Node	6.3%	2.4%
Fusion Node	6.3%	7,5%

Table 4
Ergane in terms of energy consumption.

MKF	Ergane Collector node	MAF Collector node
S1	(4.640 ± 0.248) µJ	(4.540 ± 0.244) µJ
S2	(4.810 ± 0.175) µJ	(4.516 ± 0.180) µJ
S3	(4.780 ± 0.270) µJ	(4.640 ± 0.097) µJ
S4	(4.810 ± 0.313) µJ	(4.680 ± 0.127) µJ

- If $Risk(H0/U_{\pi_i}) > Risk(H1)/(1-U_{\pi_i})$; H_0 is rejected and π_i is not appended to the joint decision set.
- If $(1-U_{\pi_i})Risk(H0) < U_{\pi_i}Risk(H1)$; H_0 is not rejected and π_i is appended to the joint decision set.

The Hypothesis test output is whether to reject H_1 meaning H_0 leads to the smaller posterior expected loss, otherwise reject H_0. Considering d_0 the output that reject H_1 and d_1 the one who rejects H_1. When d_1 is chosen and H_0 is actually the true hypothesis (type I error), then the loss is $(\sum_{j=1}^{n} C_{ji})/n$. If decision d_0 is chosen and H_1 is true (type II error), then the loss is C_{ij} meaning the cost of $\pi_i^c|D$. If the decision d_0 is chosen when H_0 is the true hypothesis or if the decision d_1 is chosen when H_1 is the true hypothesis, then the correct decision has been made and the loss is 0.

6. Resource consumption results

As a MKFH, is intended to function on a resource-constrained scenario such as WSANs. E1 was designed to assess the impact of in terms of resource consumption. It assesses in terms of memory and energy consumption, to assess

6.1. E1: Pallas in terms of resource consumption

To assess memory consumption, implementation of Pallas was deployed on Zolertia Z1 real nodes alongside Contiki OS 2.7 and the total amount of flash memory was gathered. The total consumed amount was compared with the total amount Zolertia Z1 specified on [37]. To check the amount of RAM memory consumed, Pallas run the scenario described in Section 5.1 with two applications deployed (Overhead power line and battery applications), then the total amount of RAM memory was collected. Table 1 shows the results for both flash and RAM memory consumed by Pallas.

In short, using the instant contiki 2.7 version we were able to deploy the Algorithm on real nodes and estimate its usage of RAM and flask memory. The Zolertia Z1 sensor platform, the proposed MKF in this work requires around 7% of total RAM memory and around 3% of total Flash memory for collector node. The fusion node requires around 6% of total RAM memory and around 7% of total Flash memory. It may be noted that none of the nodes exceeds the 8 kB RAM available. Nevertheless, we can conclude that in terms of memory usage, the proposed Algorithm is feasible to be employed in a real WSAN deployment.

To assess energy consumption we used cooja simulation with aid of a powertrace tool [30]. The implementation of both MAF (Section 5.6) and Pallas algorithms were deployed on simulated Zolertia Z1 nodes. Each simulation lasted 1 h and was repeated 30 times, obtaining a confidence interval of 95%. With aid of power trace tool, [30] present at Cooja simulator we were able to estimate the average node energy consumption for one cycle. Table 2 provide its result.

A battery commonly used by Zolertia is a Li-ion battery of 3.7 V and 800mAh (2,96 J) capacity. Considering that battery loses energy only from Pallas consumption and the average node energy consumption is expended every cycle, Pallas Collector node can last around 594377 cycles in S1, 649122 cycles in S2, 612456 cycles in S3 and 626587 cycles in S4.

For S1 and S2 Pallas Collector node behaviour is similar to MAF's Collector node behaviour, regardless of the probability calculation. So, Pallas overhead for S1 is about 2% and the results for S2 are similar for both algorithms. Regarding S3 and S4, Pallas has a slightly higher energy consumption 2.5% as MAF energy consumption remains the same. The major fact that explains this behaviour is that for S3 and S4 Pallas send alongside MAF's output a set of probabilities, which increase the message size for each application decision. In short, The IEEE 802.15.4 standard supports the maximum frame size up to 127 bytes including 25 bytes of MAC header and 102 bytes of payload. MAF's output has size of 3 bytes and pallas need to append the out put of a set of probabilities, which increase the message size by 4 bytes for each application. When Pallas runs with more then 24 applications the number of messages sent by Pallas doubles.

6.2. E1: Ergane in terms of resource consumption

Similarly to Pallas approach, implementation of Ergane was deployed on Zolertia Z1 real nodes alongside Contiki OS 2.7 and the total amount of flash memory was gathered and compared with the total amount for Zolertia Z1 [37]. To check the amount of RAM memory consumed, Ergane has run the application scenario described at Section 5.1. Then the total amount of RAM memory was collected. Table 3 shows the results for both flash and RAM memory consumed by Ergane.

Ethe range does not provide the overhead of storing the application's logic, the only information needed to its execution is the period of time between readings. So it is feasible to attend any number of applications. In the Zolertia Z1 sensor, Ergane requires around 6% of total RAM memory and around 3% of total Flash memory for collector node. The fusion node requires around 5% of total RAM memory and around 8% of total Flash memory. It may be noted that none of the nodes exceeds the 8 kB RAM available. So, we can conclude that the proposed Algorithm is feasible to be employed in a real WSAN deployment.

To assess energy consumption, simulated Zolertia Z1 nodes we deployed first with MAF's algorithm Section 5.6 and then with Ergane's. Each simulation lasted 1 h and was repeated 30 times, obtaining a confidence interval of 95%. Using the power trace tool [30] present at Cooja simulator [29] values of nodes energy consumption were collected and an estimation for the average node energy consumption of a cycle made. Table 4 provides its result.

As expected, for all situations Ergane Collector node behaviour is similar to MAF's and the energy consumption is the same for both algorithms. Therefore, Ergane energy consumption behaviour is compatible with the WSAN paradigm.

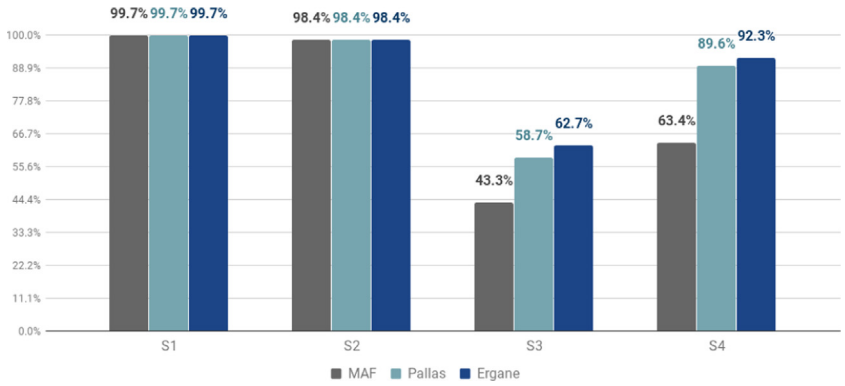

Fig. 7. Pallas accuracy results in function of Costs.

7. Accuracy results

Decision making for multiapplication WSAN scenarios may raise destructive relationships among application's decisions as contradictory or redundant outcomes, our MKFH deals with such a challenge. In a way, it allows the application's decisions integration reducing its contradictions and redundancies. As a study case for this challenge, E2 and E3 were designed to assess the impact in terms of decision making accuracy. E2 target is to provide a baseline for comparing MAF with our algorithm. Regarding E3, its objective is to assess how the MKFH enhances the decision making experience as it reaches more accurate integrations. For evaluating the integration standalone, both cases run on Cooja simulations for every situation under consideration and generated a log report for the communications among the SkN and SeNs. Those log reports were given as input for Pallas and Ergane integration algorithm setting up the Costs according to Table 5 for E2 and Tables 6 and 7 for E3. Results for Pallas were depicted in Section 7.2 and for Ergane in Section 7.3.

The initial air temperature, battery temperature and the overhead power line temperature were respectively set as 25 °C, 44 °C and 44 °C. Experiments for each situations reached a 95% confidence interval for its results and were presented at Table 5 for E2 and Tables 6 and 7 for E3. The fusion window and the interval to collect samples are set according to the most demanding application, in this case, OPLM application. Four-time slots called S1, S2, S3 and S4 were considered. Each time slot represents a particular situation submitted to the applications in time, meaning a particular event to be detected. S1 represents ideal conditions for both applications is a safe condition, where there is no need for preventive actions. S2 represents an increase in power line temperature. S3 represents an increase in the battery temperature representing a risk for battery integrity. At last, S4 represents the occurrence of an increase in both power line and battery temperature. During S1 the temperatures vary from 44 °C to 65 °C both in the Overhead power and in the battery. During S2, temperatures vary from 65 °C to 85 °C in the Overhead power line. During S3, the temperatures vary from 44 °C to 65 °C in the overhead power line and are over 140 °C in the battery. Finally, S4 temperature in the Battery is over 144 °C and over 80 °C in the overhead power line. At S3 the expected behaviour of both applications combined is considered contradictory. At S4 the expected behaviour of both applications combined is redundant.

S1 maps a condition in which the condition is ideal. For S2 only the power line is affected and the correct action is implemented by OPLM application. Regarding S3, only the battery is the one responsible for increasing the environment temperature. Also, applying OPLM and Batt designed actions together means that the transmission tower will be offline when operating on safe conditions wasting energy. The expected behaviour for this situation is one in which only the Batt application is triggered to act. S4 maps a failure condition in which both systems must be shut down by the OPLM application. Experiment E2 set the Costs to 1 and compare its results with MAF. In E3 the Costs were set to investigate how it changes the decision making process.

Table 5
MAF accuracy baseline.

Algorithm	MAF	Pallas Athena	Ergane Athena
S1	(99.7 ± 0.003)%	(99.7 ± 0.002)%	(99.7 ± 0,003)%
S2	(98.4 ± 0.010)%	(98.4 ± 0.010)%	(98.4 ± 0.008)%
S3	(43.3 ± 0.007)%	(58.7 ± 0,005)%	(62.7 ± 0.004)%
S4	(63.4 ± 0.006)%	(89.6 ± 0,003)%	(92.3 ± 0.004)%

7.1. E2: MAF accuracy baseline

Setup E2 compare our algorithms with MAF in terms of accuracy for its decision making, setting a baseline needed to assess Athena's results as the Costs changes. The results are shown in Table 5 and Fig. 7.

For S1 and S2 MAF, Pallas and Ergane function very similarly. Results show that the precision and accuracy of both algorithms are the same for S1 and S2 situations. Nevertheless, when it comes to situation S3 and S4 contradictory and redundant behaviour start to occur. The results show that our MKFH supports better integration, as well as increases substantially the accuracy on events of interest recognition, at least 15% without compromising its accuracy.

7.2. E3: Pallas in terms of accuracy

To assess Pallas accuracy, E3 provided a baseline to compare our algorithm with MAF. However, the MKFH reviews how the decision making agents weight the information on the fly to enhance integration. In a way, this reviewing leads to better event recognition, allowing more accurate decisions. In summary, as a learning component accumulates reward, a value function map the overall experience into a variable meaning the loss which is expected to occur from the deployment of action when other actions were into consideration. That way, E3 run tests for S1, S2, S3 and S4 situations under the same variation of Costs in KDS. That way, E3 evaluate how a change in the KDS is translated into precision and accuracy on the recognition of the Overhead power line and Battery events of interest. As stated earlier MAF and our algorithms were expect to function very similarly at S1 and S2 conditions. E3 showed us that for every combination of experimented Costs Pallas results are the same as MAF for S1 and S2 conditions.

Table 6 and Fig. 8 provides the results of S3 and S4, first the baseline set in E2 is provided. Lines 2 to 4 maps a change on the ratio of application's Costs, meaning only Batt's decision is expected to be right when the OPLM's decision is also under consideration. Line 5 to 7 maps a change on the ratio of application's Costs, meaning only OPLM's decision is expected to be right when the Batts' decision is also under consideration.

Regarding lines two to four, the results show that by changing the Cost's ratio to favour Batt, Pallas can better assess the environment behaviour, also reaching more suitable decisions for S3. For S4, lines

Table 6
Pallas accuracy results in function of cost ratio.

Batt/OPLM Cost	S3	S4
1/1	(58.7 ± 0,003)%	(89.6 ± 0,003)%
1/5	(58.7 ± 0,004)%	(89.0 ± 0,005)%
1/10	(73.0 ± 0,002)%	(64.7 ± 0,006)%
1/15	(73.0 ± 0,002)%	(64.7 ± 0,006)%
5/1	(58.3 ± 0,007)%	(90.2 ± 0,002)%
10/1	(42.0 ± 0,012)%	(94.2 ± 0,001)%
15/1	(42.0 ± 0,012)%	(94.2 ± 0,001)%

Table 7
Ergane accuracy results in function of Costs.

Batt/OPLM Cost	S3	S4
1/1	(62.7 ± 0,004)%	(92.3 ± 0,004)%
1/5	(67.2 ± 0,006)%	(92.1 ± 0,006)%
1/10	(78,1 ± 0,002)%	(92.1 ± 0,006)%
1/15	(78.1 ± 0,003)%	(92.3 ± 0,010)%
5/1	(61.5 ± 0,007)%	(98.2 ± 0,002)%
10/1	(60.3 ± 0,090)%	(98.2 ± 0,001)%
15/1	(60.3 ± 0,007)%	(98.2 ± 0,002)%

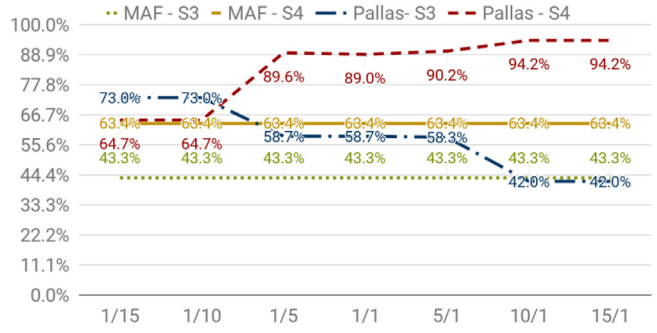

Fig. 8. Pallas accuracy results in function of Cost ratio.

five to seven, the results show that Pallas reaches more suitable decisions, by changing the Cost's ratio to favour OPLM. When compared to baseline results (line one), the learning factor explains the enhancement of around 14% in precision for S3 (line 3) and S4 (line 6). When compared directly to MAF, best case scenario indicates a decision enhancing around 30% more for S3 and S4. Worst case scenario Pallas has around 1% less precision for S3 and S4 1% more precision for S4.

Our MKFH try to decide which of two joint distributions better describes environmental changes. In situations that produced redundant or contradictory behaviour, choosing a set of action that minimizes undesirable behaviour risk. It may choose as most probable scenario a joint distribution that models the temperature as deriving from the power line or one that model de temperature as deriving from batteries. The first case, OPLM will have priority, minimizing the error from choosing Batt's actions when OPLM's actions are most desirable. The second case, Batt will have priority, minimizing the error from choosing OPLM's actions when Batt's actions are most desirable.

On Bayesian hypothesis test procedures, the priority as stated is a function of its prior function and the loss function. The prior function loses weight as new evidence comes up. Thus, the ratio of the costs is the most important variable to map the priority feature. Our learning component value function accumulates rewards to adapt its hypothesis loss function. In a way, the decision making agent who uses the test to assess the situation is learning how to decide in an uncertain context. Thus, as it revises its decisions receiving rewards from it, it accumulates knowledge regarding the decision making.

7.3. E3 — Ergane in terms of accuracy

To assess Ergane integration's behaviour, E3 evaluates how a change in the KDS affects the recognition of the Overhead power line and

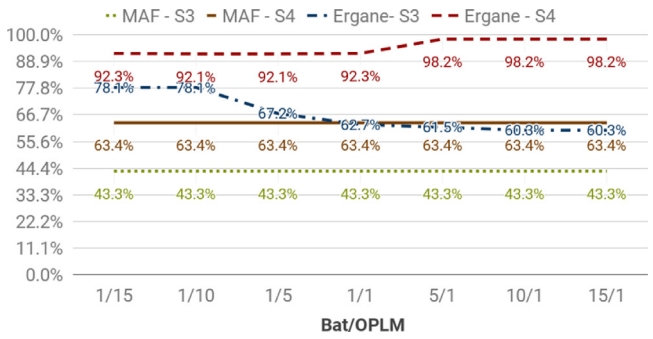

Fig. 9. Ergane accuracy results in function of Cost ratio.

Battery events of interest. MAF, Pallas and Ergane were expected to function very similarly at S1 and S2 conditions, that way E3 showed us that for every combination of experimented Costs that Ergane results are the same as MAF for S1 and S2 conditions. For S3 and S4, results were shown on Table 7 and Fig. 9.

Regarding lines two to four, the results show that by changing the Cost's ratio, Ergane can better assess the environment behaviour, also reaching suitable decisions. In a way, it increases the frequency Ergane successfully decides over S3 conditions. This behaviour change occurs at the expense of precision and accuracy on recognition of the S4 condition where OPLM decisions are more desirable. At lines four to seven, the situation turns completely. With the increase in the Costs ratio can better map the Batt behaviour, causing an increase in the frequency in which the MKFH successfully decides over S4 conditions. This behaviour change occurs at the expense of accuracy on the recognition of the S3 conditions where Batt decisions are more desirable.

When compared to baseline results (line 1), the learning factor explains the enhancement of 14% in precision for S3 (line 3) and S4 (line 6). When compared directly to MAF, best case scenario indicates a decision enhancing 35% more for S3 and S4. Worst case scenario against MAF, Ergane has around 17% more precision for S3 and S4 28% more precision for S4. When compared directly to Pallas, best case scenario indicates a decision enhancing around 37% more for S3 and S4. Worst case scenario against Pallas, Ergane has around 5% more precision for S3 and S4.

On Bayesian hypothesis test procedures, the priority as stated is a function of its prior function and the loss function. The prior function loses weight as new evidence comes up [20]. Thus, the ratio of the costs is the variable to map how the learning agent value function accumulates rewards. In a way, the decision making agent who uses the test to assess the situation is learning how to decide in an uncertain context. Thus, as it revises its decisions receiving rewards from it, it accumulates knowledge regarding the decision making and uses it to reach better decisions.

The general idea is to choose the action that leads to the smaller risk, which is a function of Cost ratio, prior probability function and likelihood function. Tables 5–7 each line carries the same cost ratio and prior distribution. Thus, Ergane benefits from an environment that has a larger pool of computational resources to model the event probability as a continuous likelihood function. As consequence, it can capture more subtle changes allowing it to use better and more complex models for event probabilities enhancing its situation assessment, as well as reaching better results on Ergane.

Indeed, the behaviour above mentioned it is due to the capability to perform the integration between these two applications. In the aforementioned scenario, for S3 and S4, the OPLM and Batt decisions are redundant or contradictory, so only one of them can respond successfully to the problem without non-desirable energy consumption. The learning agent allows Ergane to summarize into a score each

previous decision making experience. This feature in addition to priori allowing our algorithms to adapt part of its hypothesis to reach a Bayesian hypothesis test procedure more suitable to assess a situation.

8. Results summary

The conducted experiments focused on assessing if our algorithms were compatible with the WSAN scenario, their ability to recognize events of interest in a resource-constrained environment and how the integration among applications work as the learning agent revise its hypothesis. Experiments were made on two possible implementations of our MKFH — Pallas and Ergane.

Memory consumption results for Pallas showed that it is compatible with the scenario under consideration. However, as the number of applications increase memory consumption also rise. Regarding Ergane, its centralized nature makes it unsuited for critical applications. Its memory consumption remains unchanged for any number of applications. Energy consumption results for Pallas showed that for the scenario under consideration, its energy consumption is similar to MAF. However, a simple analysis of its messaging behaviour is sufficient to show that its energy cost increases with the number of applications deployed. Regarding Ergane, its energy consumption is identical to MAF's and does not increase with the number of applications.

The second assessment set a baseline to evaluate integration behaviour. Pallas, Ergane and MAF were compared due to its accuracy results without any intervention by the learning agent, differing only by a probabilistic data fusion. Results For situation S1 and S2, MAF and the behaviour of our algorithms were the same. When compared to MAF, Pallas has increased in 15% the accuracy on recognition of the power line overload event on situation S3. For the same situation Ergane's results points for a gain of 20% in accuracy over MAF.

The third assessment evaluates the integration behaviour regarding changes in the ratio of the costs. Results showed a great impact on the integration process due to changes in cost ratio, which highlight the ability of selectively applying knowledge from previous experience to enhance accuracy. Regarding situation S1 and S2 Pallas and Ergane behaviour derive only from data fusion, so the results for Pallas and Ergane were the same as MAF. For situation S3, the learning factor explains the enhancement of around 15% on Pallas accuracy as the cost of the OPLM application increase, which is the desired effect for this situation. So, for the best-case scenario, Pallas has an increase of 30% of accuracy when compared to MAF. The worst-case scenario occurs when the cost of the right decision increase when the cost of the battery application increase Pallas has shown a loss of 1% of accuracy if compared to MAF. S4 showed similar results, the cost ratio favouring the right application resulted in an enhancement of 30% of accuracy. When the cost ratio favoured the wrong application, Pallas accuracy was only 1% greater then MAF's. Regarding Ergane, when compared to MAF the best case scenario showed that Ergane accuracy increase around 35% S3 and S4. At the worst-case scenario, Ergane's accuracy was around 17% better for S3 and 28% better for S4.

9. Final remarks

This study presented a multisensor knowledge fusion heuristic (MKFH), which combines data fusion and knowledge fusion to allow changes to the integration hypothesis to occur on the fly. Our MKFH aims at integrating a multilevel decision making process compatible with resource-constrained scenarios which produce large amounts of data in lower semantic levels. It uses data fusion to fuel applications decision making, assessing environmental changes, adding meaning to it and enhancing the semantic level of gathered data. Knowledge fusion manages how consistent the environment feedback is due to an overall reasoning process. By mapping and introducing changes on decision hypothesis, It extracts and selectively applies knowledge to enhance the overall decision making. To allow combining data fusion and knowledge fusion, a contextual information model was proposed. This model captures particularities on how different applications assess an environmental change storing it on a knowledge data structure. In a way, the contextual information represents a fact, from which different application's decisions correlate.

Following Schmidt [38] classification for decision making level. Application's decision is on the operational level, aiming to reach decisions that suit local needs. Integration is on the tactical level, deciding on a more complex picture and dealing with conflict. In this context, our MKFH provides a way to integrate both by extracting and transferring knowledge. In a way, a knowledge graph instance map knowledge about conflict and its history how it has changed. Therefore, multiple instances could be compiled to picture the long term system behaviour — the decision experience and to provide insight for the strategic level.

We used a Smart grid scenario found in [23]. Experiments have shown that first introduced an algorithm, Pallas Athena is compatible with the scenario under consideration. However, to allow Pallas to deal with critical applications, sensor nodes must be aware of the application's decisions causing overheads on memory and energy consumption. When compared to MAF for setting the baseline, has increased in 15% the accuracy on recognition of the power line overload. Also, as a semi-centralized approach, it allowed knowledge extraction and integration of critical and non-critical applications. On the scenario when all applications are set as critical, its behaviour is identical to its data fusion method plus knowledge extraction. Regarding Ergane, memory and energy consumption was unchanged by the number of applications. When compared with MAF for setting the baseline, its results were 20% better in terms of accuracy. Third assessment results for both algorithms showed that changes in the ratio of the costs had a great impact on integration.

Both algorithms data fusion and knowledge fusion components are completely independent, so changing a single component does not affect the whole algorithm. However, Pallas preprocessing component is not completely decoupled from its applications. This causes overheads in terms of memory consumption and threatens the viability of a decentralized approach. Ergane allows complete decoupling of the applications from the infrastructure, also completely decouples data fusion from knowledge fusion. Providing insight for complex strategies regarding the management of energy resources

As Future research, we highlight: (1) study how different data fusion algorithm could support our MKFH; (2) research how to model knowledge extraction and transfer and its behaviour when supported by single and multi-agent approaches; (3) investigate MKFH core components under a decentralized approach; (4) apply the MKFH for a wider spectrum of IoT applications; (5) research how to create manageable IoT knowledge bases supported by our MKFH:

(1) Data fusion based algorithms are a key component for our MKFH. Thus, it is important to evaluate the impact on the accuracy and resource consumption when supported by different data fusion algorithm. Replace MAF with other approaches may make some trade-off more suitable for some IoT applications. New implementation can be made to assess the impact of changing the data fusion component, which should contribute to more suitable applications design for the IoT paradigm [2].
(2) A key component, reinforcement learning was designed to be as simple as possible. That way, more complete and complex reinforcement learning approaches may be implemented [9]. The evaluation of the MKFH behaviour, when supported by different reinforcement learning approaches, is challenging and has many opportunities. Markov model can be used for a certain reinforcement learning schemes, as well as Q-learn schemes [39–41]. Analytical models can be developed to obtain a more in-depth understanding of learning [42,43]. Single-agent learning, where sink nodes or sensor nodes learn independently from the environment or Multi-agent, where they learn cooperatively could

be investigated [44]. Multi-agent RL is a potential technique to improve performance, however information exchange increases overheads, so the trade-off needs to be considered [45].

(3) Pallas and Ergane were designed in a centralized manner, investigating its core components under a decentralized approach could expose interesting features [46]. Also, allowing management of WSANs constrained resources more efficiently with more refined strategies [25]. As consequence, increasing WSAN's lifetime and supporting faster and more accurate decisions.

(4) Applying over different scenarios under the IoT paradigm. The IoT paradigm is vast in terms of complex applications, understanding how our MKFH performs under different IoT use cases should incrementally strengthen its generality [18,19]. Newer use cases for the MKFH, using real nodes or real data sets could be a way to polish it into knowledge management for a wider spectrum of applications. In a way, contributing to creating large knowledge repositories mapping how different application performs under a different context.

(5) Finally, investigate the use of MKFH as the main component in intelligent fusion systems to support refinement of vast knowledge bases and the attainment knowledge graphs [18]. As consequence, allow mapping the fit of an application to a specific scenario, as well as support learning newer and more efficient strategies hidden under the correlations among different applications.

CRediT authorship contribution statement

Gabriel Martins: Writing - original draft, Conceptualization, Formal analysis. **Sergio Guedes de Souza:** Writing - review & editing, Methodology. **Igor Leão dos Santos:** Writing - review & editing. **Luci Pirmez:** Supervision. **Claudio M. de Farias:** Writing - review & editing, Supervision, Methodology.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Y. Recommendation, 2060 Overview of Internet of Things, ITU-T, Geneva (2012).
[2] E.F. Nakamura, A.A. Loureiro, A.C. Frery, Information fusion for wireless sensor networks: Methods, models, and classifications, ACM Comput. Surv. 39 (3) (2007) 9.
[3] B. Khaleghi, A. Khamis, F.O. Karray, S.N. Razavi, Multisensor data fusion: A review of the state-of-the-art, Inf. Fusion 14 (1) (2013) 28–44.
[4] M.E. Jennex, S.E. Bartczak, A revised knowledge pyramid, Int. J. Knowl. Manag. (IJKM) 9 (3) (2013) 19–30.
[5] R.L. Ackoff, From data to wisdom, J. Appl. Syst. Anal. 16 (1) (1989) 3–9.
[6] G. Martins, C.M. de Farias, L. Pirmez, Athena: A knowledge fusion algorithm for the internet of things, in: Proceedings of the 14th ACM International Symposium on QoS and Security for Wireless and Mobile Networks, ACM, 2018, pp. 92–99.
[7] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, W. Zhang, Knowledge vault: A web-scale approach to probabilistic knowledge fusion, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2014, pp. 601–610.
[8] J. Shi, H. Gao, G. Qi, Z. Zhou, Knowledge graph embedding with triple context, in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, ACM, 2017, pp. 2299–2302.
[9] S. Richard, Sutton and Andrew G. Barto, "Reinforcement Learning: An Introduction", MIT Press, 1998.
[10] A. Smirnov, T. Levashova, N. Shilov, Patterns for context-based knowledge fusion in decision support systems, Inf. Fusion 21 (2015) 114–129.
[11] X.L. Dong, E. Gabrilovich, G. Heitz, W. Horn, K. Murphy, S. Sun, W. Zhang, From data fusion to knowledge fusion, Proc. VLDB Endow. 7 (10) (2014) 881–892.
[12] M.I. Akhlaghi, S.V. Sukhov, Knowledge fusion in feedforward artificial neural networks, Neural Process. Lett. 48 (1) (2018) 257–272.
[13] A. Preece, K. Hui, A. Gray, P. Marti, T. Bench-Capon, Z. Cui, D. Jones, KRAFT: An agent architecture for knowledge fusion, Int. J. Coop. Inf. Syst. 10 (01n02) (2001) 171–195.
[14] E.U. Kriegel, S. Pfennigschmidt, H.G. Ziegler, Practical aspects of the use of a knowledge fusion toolkit in safety applications, in: 2013 IEEE Eleventh International Symposium on Autonomous Decentralized Systems, ISADS, IEEE, 2013, pp. 1–4.
[15] R. Pochampally, A. Das Sarma, X.L. Dong, A. Meliou, D. Srivastava, Fusing data with correlations, in: Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, ACM, 2014, pp. 433–444.
[16] K. Devlin, Confronting Context Effects in Intelligence Analysis: How Can Mathematics Help, Center for the Study of Language and Information, Stanford University, 2005.
[17] P.S.M. Dos Santos, G.H. Travassos, Scientific knowledge engineering: a conceptual delineation and overview of the state of the art, Knowl. Eng. Rev. 31 (2) (2016) 167.
[18] N. Zhang, H. Chen, X. Chen, J. Chen, Semantic framework of internet of things for smart cities: Case studies, Sensors 16 (9) (2016) 1501.
[19] M. Mohammadi, A. Al-Fuqaha, M. Guizani, J.-S. Oh, Semisupervised deep reinforcement learning in support of IoT and smart city services, IEEE Internet Things J. 5 (2) (2018) 624–635.
[20] M.H. DeGroot, M.J. Schervish, Probability and statistics, Pearson Education, 2012.
[21] C. de Farias, A Framework for Developing Smart Space Applications Using Shared Sensor Networks (Ph.D. thesis), Universidade Federal do Rio de Janeiro - UFRJ, 2014.
[22] G. Aquino, L. Pirmez, C.M. de Farias, F.C. Delicato, P.F. Pires, Hephaestus: A multisensor data fusion algorithm for multiple applications on wireless sensor networks, in: 2016 19th International Conference on Information Fusion, FUSION, IEEE, 2016, pp. 59–66.
[23] C.M. de Farias, L. Pirmez, F.C. Delicato, I.L. Dos Santos, A.Y. Zomaya, Information fusion techniques applied to shared sensor and actuator networks, in: 37th Annual IEEE Conference on Local Computer Networks, IEEE, 2012, pp. 188–191.
[24] J.A. Momoh, Smart Grid: Fundamentals of Design and Analysis, vol. 63, John Wiley & Sons, 2012.
[25] I.L. Santos, L. Pirmez, F.C. Delicato, L.F.R. da Costa Carmo, Ensuring energy efficiency of power quality applications in smart grids through a framework based on wireless sensor and actuator networks, in: 2015 International Wireless Communications and Mobile Computing Conference, IWCMC, IEEE, 2015, pp. 763–768.
[26] S.A. Gal, M.N. Oltean, L. Brabete, I. Rodean, M. Opincaru, On-line monitoring of OHL conductor temperature; live-line installation, in: 2011 IEEE PES 12th International Conference on Transmission and Distribution Construction, Operation and Live-Line Maintenance, ESMO, IEEE, 2011, pp. 1–6.
[27] M. Schlapfer, P. Mancarella, Probabilistic modeling and simulation of transmission line temperatures under fluctuating power flows, IEEE Trans. Power Deliv. 26 (4) (2011) 2235–2243.
[28] A. Dunkels, B. Gronvall, T. Voigt, Contiki-a lightweight and flexible operating system for tiny networked sensors, in: 29th Annual IEEE International Conference on Local Computer Networks, IEEE, 2004, pp. 455–462.
[29] F. Österlind, A Sensor Network Simulator for the Contiki OS, SICS Research Report, Swedish Institute of Computer Science, 2006.
[30] A. Dunkels, J. Eriksson, N. Finne, N. Tsiftes, Powertrace: Network-Level Power Profiling for Low-Power Wireless Networks, Swedish Institute of Computer Science, 2011.
[31] S. Xiong, J. Li, M. Li, J. Wang, Y. Liu, Multiple task scheduling for low-duty-cycled wireless sensor networks, in: 2011 Proceedings IEEE INFOCOM, IEEE, 2011, pp. 1323–1331.
[32] C. Farias, L. Pirmez, F. Delicato, L. Carmo, W. Li, A.Y. Zomaya, J.N. de Souza, Multisensor data fusion in shared sensor and actuator networks, in: Information Fusion (FUSION), 2014 17th International Conference on, IEEE, 2014, pp. 1–8.
[33] I. ISO, 5725-1: 1994, Accuracy (Trueness and Precision) of Measurement Methods and Results-Part 1: General Principles and Definitions, International Organization for Standardization, Geneva, 1994.
[34] E. Prenesti, F. Gosmaro, Trueness, precision and accuracy: a critical overview of the concepts as well as proposals for revision, Accredit. Qual. Assur. 20 (1) (2015) 33–40.
[35] I.L. Dos Santos, L. Pirmez, É.T. Lemos, F.C. Delicato, L.A.V. Pinto, J.N. de Souza, A.Y. Zomaya, A localized algorithm for structural health monitoring using wireless sensor networks, Inf. Fusion 15 (2014) 114–129.
[36] N. Metropolis, S. Ulam, The Monte Carlo method, J. Amer. Statist. Assoc. 44 (247) (1949) 335–341.
[37] W. Zolertia, Platform, Z1 Datasheet, March, 2010.
[38] G. Schmidt, W.E. Wilhelm, Strategic, tactical and operational decisions in multinational logistics networks: a review and discussion of modelling issues, Int. J. Prod. Res. 38 (7) (2000) 1501–1523.
[39] Y. Chu, P. Mitchell, D. Grace, Reinforcement Learning Based ALOHA for Multi-Hop Wireless Sensor Networks with Informed Receiving, IET, 2012.
[40] Y. Chu, P.D. Mitchell, D. Grace, ALOHA And q-learning based medium access control for wireless sensor networks, in: 2012 International Symposium on Wireless Communication Systems, ISWCS, IEEE, 2012, pp. 511–515.
[41] Y. Chu, S. Kosunalp, P.D. Mitchell, D. Grace, T. Clarke, Application of reinforcement learning to medium access control for wireless sensor networks, Eng. Appl. Artif. Intell. 46 (2015) 23–32.

[42] M. Wiering, M. Van Otterlo, Reinforcement learning, Adapt. Learn. Optim. 12 (2012) 3.
[43] B. Kiumarsi, K.G. Vamvoudakis, H. Modares, F.L. Lewis, Optimal and autonomous control using reinforcement learning: A survey, IEEE Trans. Neural Netw. Learn. Syst. 29 (6) (2018) 2042–2062.
[44] M.I. Khan, B. Rinner, Energy-aware task scheduling in wireless sensor networks based on cooperative reinforcement learning, in: 2014 IEEE International Conference on Communications Workshops, ICC, IEEE, 2014, pp. 871–877.
[45] M.I. Khan, Resource-aware task scheduling by an adversarial bandit solver method in wireless sensor networks, EURASIP J. Wireless Commun. Networking 2016 (1) (2016) 10.
[46] I.L.d.S. Santos, On the Virtualization and Resource Allocation in the Cloud of Sensors (Ph.D. thesis), Universidade Federal do Rio de Janeiro - UFRJ, 2017.