CONTENTS

IMCF: The IoT Meta-Control Firewall for Smart Buildings

Soteris Constantinou
University of Cyprus
2109 Nicosia, Cyprus
constantinou.sotiris@cs.ucy.ac.cy

Antonis Vasileiou
University of Cyprus
2109 Nicosia, Cyprus
avasil06@cs.ucy.ac.cy

Andreas Konstantinidis
Frederick University
1036 Nicosia, Cyprus
com.ca@frederick.ac.cy

Panos K. Chrysanthis
University of Pittsburgh
Pittsburgh, PA 15260, USA
panos@cs.pitt.edu

Demetrios Zeinalipour-Yazti
University of Cyprus
2109 Nicosia, Cyprus
dzeina@cs.ucy.ac.cy

ABSTRACT

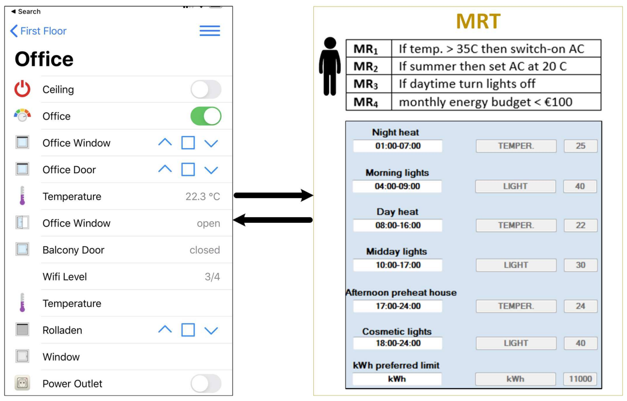
In this demonstration paper, we present an innovative *IoT Meta-Control Firewall* (*IMCF*), which allows users to schedule their IoT devices in smart buildings (e.g., heating, cooling, lights) in order to reach some long-term energy consumption objective (e.g., consume less than 400 kWh in December) while, at the same time, retaining high levels of user convenience (comfort). IMCF internally deploys an AI-inspired Energy-Planner (EP) algorithm that exploits domain-specific operators to balance the trade-off between convenience and energy consumption. Our framework then filters the rules of users in a way that these do not conflict with the long-term objectives (i.e., like a network firewall). We demonstrate IMCF using a prototype system we have developed in the Laravel PHP web framework using the open Home Automation Bus (OpenHAB), the Linux crontab daemon and Anyplace for building modeling. In our demonstration scenario, attendees will be able to observe the execution and benefits of IMCF on a graphical dashboard using pre-configured or custom-made Meta-Rule-Table profiles.

1 INTRODUCTION

Internet of Things (IoT) refers to a large number of physical devices being connected to the Internet that are able to “see", “hear", “think", “react", perform tasks, as well as communicate with each other using open protocols [8]. According to Gartner[1], it is expected that the number of IoT devices per house will increase to more than 500 smart devices by 2022. Many IoT devices also enable the execution of *Rule Automation Workflows (RAW)*, which span from simple predicate statements to procedural workflows capturing a smart actuation pipeline in tools like IFTTT [5], controlling Philips Hue lights, BMW i3 EVs or Daikin A/Cs [7][4], Apilio.io, or Apple Automation.

RAW aim to meet the convenience level of users under specific conditions (e.g., *“warm house to 22°C if cold or preheat Electric Vehicle when approaching”*). In the simplest case, a user expresses preferences manually through a vendor-specific smartphone app / integrated app (e.g., see Fig. 1). This process requires continuous attention by custodians, making it a cumbersome process that generates erroneous executions and that clearly calls for more automated (i.e., *“smarter”*) approaches.

[1]Gartner Inc., URL: https://tinyurl.com/ycrxsmy6



Ceiling	off
Office	on
Office Window	
Office Door	
Temperature	22.3 °C
Office Window	open
Balcony Door	closed
Wifi Level	3/4
Temperature	
Rolladen	
Window	
Power Outlet	off

MR_1	If temp. > 35C then switch-on AC
MR_2	If summer then set AC at 20 C
MR_3	If daytime turn lights off
MR_4	monthly energy budget < €100

Night heat	01:00-07:00	TEMPER.	25
Morning lights	04:00-09:00	LIGHT	40
Day heat	08:00-16:00	TEMPER.	22
Midday lights	10:00-17:00	LIGHT	30
Afternoon preheat house	17:00-24:00	TEMPER.	24
Cosmetic lights	18:00-24:00	LIGHT	40
kWh preferred limit	kWh	kWh	11000

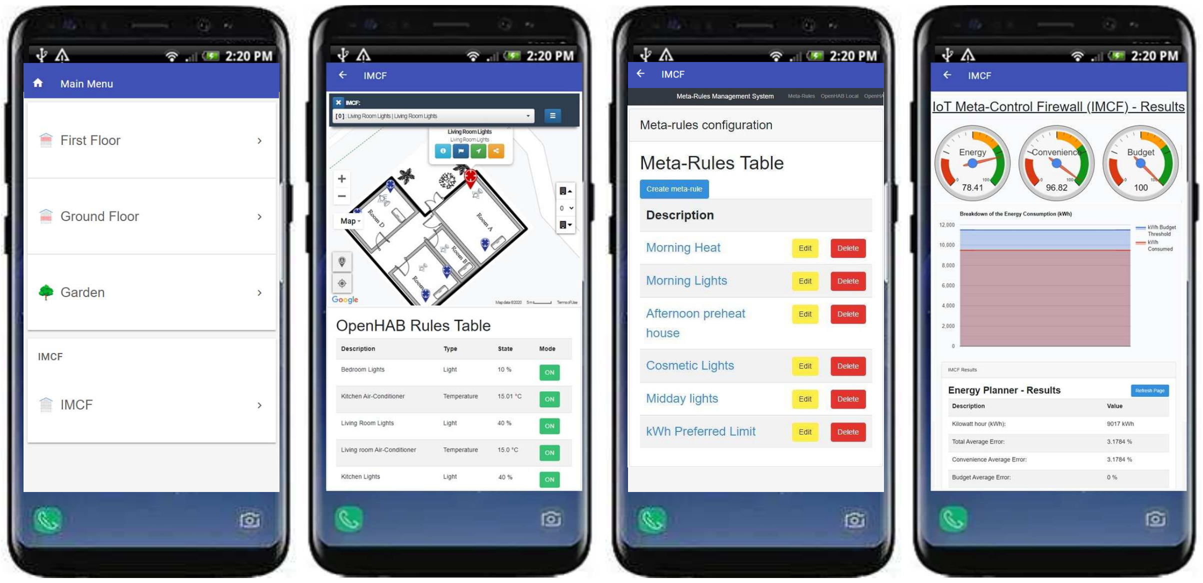
Figure 1: The openHAB bridge gives the privilege to users to adapt a Meta-Rule-Table (MRT) profile as necessary

One of the most straightforward approaches to achieve a smarter RAW is to utilize the so-called *trigger-action* model. Users control the behavior of an IoT by specifying triggers (e.g., *“if it is sunny outside”*) and their resultant actions (e.g., *“turn off the lights”*). Because of its conceptual simplicity, the trigger-action model (a.k.a. Event-Condition-Action) has attracted significant attention with ifttt.com (*“If This Then That”*) becoming one of the first large-scale deployments. Services like Apilio expanded the expressiveness of the RAW with Boolean predicates (e.g., conjunctions) and Apple Automation [3] even introduced procedural programming constructs, like variables, while loops, if statements and functions to advance RAW actuations.

However, none of the above RAW technologies enables individuals or group of users to express their convenience (comfort) preferences while achieving some long-term objective. Similarly, prior research [6] was mainly concerned with improving comfort levels of HVAC system but not long-term energy planning targets. In our scenario, the long-term objective relates to *energy consumption* (e.g., in kWh), which is motivated by European's Commission calls for a climate-neutral Europe by 2050[2]. Particularly, we aim to consume energy more intelligently and within margins we define as persons or group of users.

In this demo we present an innovative system, coined the *IoT Meta-Control Firewall (IMCF)* [2], which aims to fill the gap of manual RAW tuning to reach the energy consumption targets. The user (or group of users) start out by defining a vector of RAW rules, dubbed *MRT*, and an *Energy Consumption Profile*, dubbed *ECP*. The high-level objective is to identify among all MRT rules

[2]EU 2050 long-term strategy, `https://tiny.cc/9wu8iz`



OpenHAB Rules Table

Description	Type	State	Mode
Bedroom Lights	Light	10 %	ON
Kitchen Air-Conditioner	Temperature	15.01 °C	ON
Living Room Lights	Light	40 %	ON
Living room Air-Conditioner	Temperature	15.0 °C	ON
Kitchen Lights	Light	40 %	ON

Meta-Rules Table

Description		
Morning Heat	Edit	Delete
Morning Lights	Edit	Delete
Afternoon preheat house	Edit	Delete
Cosmetic Lights	Edit	Delete
Midday lights	Edit	Delete
kWh Preferred Limit	Edit	Delete

Energy Planner - Results

Description	Value
Kilowatt hour (kWh):	9017 kWh
Total Average Error:	3.1784 %
Convenience Average Error:	3.1784 %
Budget Average Error:	0 %

Figure 2: IMCF Graphical User Interface: Integration of the IMCF Software Library in the openHAB Home Automation Stack. From left to right: a) Interactive and Automated Menu; b) Dashboard for smart space current state linked with Anyplace Viewer; c) Meta-Rule-Table Configurator; and d) IMCF Results.

the ones that must be dropped so that the user stays within the desired energy budget according to the *ECP* history (e.g., consume less than 400 kWh in December). For this purpose, it utilizes an intelligent search algorithm, coined *EP (Energy Planner)*, which goes over the exponentially large search space of $\sum_{r \leq n} r$-combinations (where $n = |MRT|$), yielding quickly the rules to be dropped. Particularly, IMCF is composed of an intelligent energy amortization process and an AI algorithm for balancing the trade-off between convenience and energy consumption, and satisfying the RAW pipelines of users in a way that these do not conflict with the long-term objectives.

Considering several system implications, IMCF can be implemented in various ways such as the following: (i) a *cloud meta-service*, which guides the IFTTT *cloud service* that in turn controls a *local controller* (e.g., Linux, as in the case of OpenHab), which controls the sensors/actuators (e.g., A/C unit); or (ii) a *local controller* (filtering rules out at the network level on the local controller).

2 THE IOT META-CONTROL FIREWALL (IMCF) OVERVIEW

In this section, we describe a prototype system we have developed for IMCF using the open Home Automation Bus (OpenHAB)[3], the Linux crontab daemon, Anyplace for building modeling [1], as well as the Laravel PHP web framework following the model–view–controller architectural pattern.

We start out with a discussion of the system architecture, followed by the IMCF algorithm, and then describe the Graphical User Interface (GUI) we have developed. The GUI integrates directly into OpenHAB's mobile and web Panel view for both interactive management of IoT and automated management of Energy-aware MRT pipelines using the EP described below.

2.1 System Architecture

Our system architecture comprises of the following components: (i) a full-fledge local controller implemented inside the openHAB stack, which is a smart home management software; and (ii) IMCF, which is the software system that encapsulates the complete application logic of the energy management stack we propose along with the respective user interfaces.

Local Controller (LC): is a java-based system that can be installed on a micro device, like a Linux Raspberry PI, running on the local network of a user. The LC will be in direct communication with the IoT devices (i.e., *Things (TG)*) to instruct them based on the preferences registered by a user. A user will typically download the openHAB smartphone *application (APP)*, for iOS or Android, and interact with TG through LC. For the implementation of LC we decided to extend the openHAB stack, which is a vendor and technology-agnostic open source automation software for smart home that provides a rich ecosystem of bridges through which a user can interact directly with IoT devices (e.g., Daikin Smart A/C, Phillips HUE lights) both locally and remotely. This gives us the benefit to achieve maximum IoT market compatibility as the integration of IoT is always an immense challenge.

To realize the operation of LC consider, for example, a user inside his smart space that uses an APP to increase the temperature of an A/C from 21 to 25 degrees Celsius (see Figure 2a-b). This manual interaction goes directly to LC that eventually communicates with TG (on older units this is typically refers to unencrypted http communication channels, either http querystring or in some cases JSON web 2.0 interactions). When a user's APP is outside a smart space, the network firewall and Network Address Translation (NAT) will obviously not let this user interact with LC. As such, the user's APP connects to the *Cloud Controller (CC)*, which is a server on the public Internet that communicates and controls LC remotely. The complete picture can tentatively be complemented by a *Cloud Meta-Controller (CMC)*, like IFTTT, which can enable the user to configure and run various custom rules. CMC would in this case interact with CC that

[3]OpenHAB, `https://openhab.org`

Table 1: Evaluating our system prototype with respect to Energy Consumption (F_E) and Convenience Error (F_{CE})

Time Duration	Energy Consumption (F_E)	Convenience Error (F_{CE})
Week	130.64 kWh	2.35%

Table 2: Individual Resident Convenience Error (F_{CE})

Users	Convenience Error (F_{CE})
Father	0.8006%
Mother	0.7899%
Daughter	0.7595%

would in turn interact with LC that would eventually interact with TG, all under the *manual* control of the user APP.

The IMCF Component: is a software extension to LC we have implemented to enable the adaptation of convenience preferences to meet the long-term energy planning targets of individuals or group of individuals. It has been developed in a way that encapsulates the implementation of the EP algorithm but also the GUI and storage necessary to allow the user to interact with the system. EP is implemented as a JAVA library, which takes the user configurations from a local MariaDB persistency layer. The storage layer is populated by the user using the APP, which has been configured in a way to integrate seamlessly the *MRT* rule definition process through a web-based GUI (see Figure 2c,d). The GUI code is written in the Laravel PHP web framework, as well as JavaScript and HTML. The GUI code execution relies on a web-server supporting PHP while for the IMCF EP library a cron job daemon is assumed (available on Linux) that reliably invokes the Energy Planning in fixed time intervals (e.g., every few minutes). In case devices have to be turned on or off, the IMCF system has the following options in our system:

- *Binding-mode*, where IMCF exploits the rich ecosystem of bridges available on the openHAB open source project to interact with local devices. We use this as the default mode, as it allows our platform to scale up to a wide spectrum of IoT devices.
- *Extended mode*, where IMCF implements locally the custom instructions for enabling and disabling the various TG devices in the smart space of a user.
  Given that many of the IoT communications are unencrypted, this can be easily captured by deep packet analyzers like Wireshark. Moreover, to avoid any additional CMC, CC or LC interactions with TG, we also configure the LC network firewall with the `iptables` command to disable TCP flows to designated TG devices on the local network. In this case, IMCF works actually as a real network firewall by blocking all outgoing traffic from LC to TG.

Case scenario: We have deployed an instance of our real prototype system for a family of three persons for one week. Particularly, we allowed each person to configure their personal preferences using the Mobile APP that interacts with an IMCF-LC node on a Linux VM on our datacenter described earlier. Particularly, each individual resident entered approximately three different meta-rules according to their personal preferences. One of them have set the weekly energy consumption (kWh) limit to 165kWh. This results in configuration data of approximately 65 bytes / user stored in the MariaDB persistency layer. In order to measure the environmental parameters (i.e., temperature, light) we use data from the open weather forecast API. We measure the performance of the proposed EP framework in regards to Energy Consumption (F_E) and Convenience Error (F_{CE}). The F_E and F_{CE} results for our evaluation are summarized in Table 1. In respect to F_{CE} our observation is that EP is indeed an efficient approach for retrieving great user satisfaction, as it performs in 4 seconds on average with $F_{CE} \approx 2.35\%$. Table 2 demonstrates for each individual resident their own Average Convenience Error values in respect with their configured meta-rules, showing both a consistent and high satisfaction close to 99.2% for all residents. Another observation is that $F_E \approx 130.64$ kWh is within the preferred budget limit as pre-configured by the user, and the system behaves correspondingly to what we observed in the simulations. Please note, this particular framework can be equally utilized in various cases such as CO_2 emission deduction, or any other scenario type that requires a planning to conserve some kind of resources.

2.2 The *IMCF* algorithm

The *IMCF* algorithm is composed of two subroutines: (i) the *Amortization Plan (AP)*; and the (ii) the *Energy Plan (EP)*. The amortization plan is responsible for calculating the maximum energy budget constraint (coined E_p) through a pre-selected amortization formula. Then an artificial intelligence approach is executed every t seconds (e.g., hourly, daily, monthly, yearly preference) over a time period p (i.e., the complete duration of the execution) for generating an energy plan solution s^* for optimizing the Convenience Error

$$\min F_{CE} = \sum_{k=1}^{t}\left(\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{D} ce_j(MR_i)\right), \tag{1}$$

where ce_j is the difference between the desired output value $\Omega_i^j \in \Re$ of a rule set by a user (temperature or light intensity level) and the actual value $O_i^j \in \Re$ set by the controller, given by: $ce = |\Omega_i^j| - |O_i^j|$.
Subject to satisfying the Energy Consumption $F_E(s^*) \le E_p$, where:

$$F_E = \sum_{k=1}^{t}\left(\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{D} e_j(MR_i)\right), \tag{2}$$

E_p is total available energy budget for the complete period p during which the execution of our algorithm takes place, N is the total number of meta-rules, D the set of all IoT devices and e_j is the energy consumption of device j given the action defined by output O_i^j of meta-rule MR_i, given by:

$$E = \begin{cases} e_j, & \text{if } O_i^j \text{ is executed} \\ 0, & \text{otherwise} \end{cases},$$

where e_j is the energy cost of device j for MR_i.

Amortization Plan (AP) Algorithm. The $AP()$ subroutine is initially executed for calculating the energy budget constraint E_p, subject to a monthly residence *Energy Consumption Profile ECP*. There are several amortization strategies that can be used, such as Linear Amortization Formula, Balloon Linear Amortization Formula, and ECP-based Amortization Formula. Given that our approach requires no training data and only a primitive MRT preference profile, this can be easily integrated in smart actuations platforms.

Energy Plan (EP) Algorithm. An energy plan solution is a vector $s =< s_1, \ldots, s_N >$. A vector component s_i represents a meta-rule in the meta-rule-table MRT, where $s_i = 0$ means ignoring meta-rule at position i of table MRT and $s_i = 1$ means adopting meta-rule at position i. We have adopted a hill-climbing algorithm, an iterative local search heuristic, which doesn't require a learning history (like respective Machine Learning techniques), doesn't require a target function (e.g., like A*) and is straightforward to be implemented in a resource-constraint setting like local smart controllers (e.g., Raspberry). At the beginning of the local search heuristic an initial solution s^* is developed that will specify the initial state of the algorithm either randomly or deterministically. For the optimization step, a hill-climbing local search heuristic is utilized for local optimization with neighborhoods that involve changing up to k components of the solution, which is often referred to as k-opt. Each solution s is evaluated using the performance metrics F_E and F_{CE}. A solution s is considered better and replaces the current best solution s^* if $(F_E(s) \leq E_p)$ && $(F_{CE}(s) < F_{CE}(s^*))$. The energy planner stops when τ_{max} iterations are completed. Alternatively, the algorithm can iterate until $\nexists s | F_{CE}(s) < F_{CE}(s^*)$.

2.3 Graphical User Interface (GUI)

Our prototype GUI provides all the functionalities for a user participating in IMCF. The GUI is divided into a Meta-Rule-Table interface and the OpenHAB Rules Table, respectively as shown in Figure 2b-c. The Meta-Rules interface prompts users to define kWh preferred limit, temperature and light values for any configured time slots. The OpenHAB Rules Table records are retrieved through the OpenHAB Rest API system consisted of smart device sensor measurements installed and pre-configured in a building. These rule combinations are used by the AI Energy Planner algorithm to satisfy the user needs, while keeping the balance between convenience and energy consumption.

At a high level, our GUI enables the following functions: (i) record OpenHAB item measurements/values on local storage and present those on a table; (ii) configure various meta-rules in regards of kWh limit, temperature and light values; (iii) operate IMCF framework and get an efficient execution considering user satisfaction along with balanced F_{CE} and F_E.

3 DEMONSTRATION SCENARIO

During the demonstration, the attendees will be able to appreciate the key components in IMCF, as well as the adaptability and the performance of our propositions (see Figure 2).

3.1 Demo Artifact

We have implemented a prototype of IMCF as a standalone program that is loaded on a Linux-based local controller. Particularly, we implemented a graphical user interface in the Laravel PHP web framework, following the MVC architectural pattern where a user has the privilege to upload a MRT profile that is stored on the filesystem of the Linux device. A cron job has been programmed in order to initiate our energy planner every few seconds. Every time our program runs, it decides whether certain local IPs have to be banned to satisfy user's profile by interacting with the Linux firewall using the iptables commands.

3.2 Demo Plan

The conference attendees will have the opportunity to interactively engage with the OpenHAB and MRT user profile website by setting up configurations through a smartphone. We will preload a variety of synthetic and web-accessible rules to the MRT user profile website back-end. The loaded rules will capture the structure and needs of real residential data and will be very useful to visually demonstrate how the IMCF algorithm works in real time through the OpenHAB application.

The main objective of our intelligent algorithm is to identify among an exponentially large search space of MRT rule combinations the ones that must be dropped so that the user stays within the desired energy budget. In order to present the benefits of our propositions to the attendees, we will provide visual cues that will enable the audience to understand the performance benefits (i.e., CPU time) and the negligible reduction in energy consumption and convenience error we have observed during the experiments.

We will also provide conference attendees the opportunity to create custom MRT profiles through the website. Our hypothesis is that many data engineering researchers and practitioners would feel more comfortable to formulate MRT profile predicates, as opposed to be limited within the boundaries of well-defined MRT templates provided. We will provide participants the possibility to upload an actual building plot through Anyplace. The *OpenHAB* interface will allow the attendees to rapidly visualize the result-sets on a smartphone, using fancy charts (pie, bar, etc.) when the rules get enabled or disabled by the proposed firewall, respectively. Our particular aim here will be to describe how the IMCF structure, residing on the OpenHAB, will be accessible to enable/disable rule services.

4 FUTURE WORK

In the future, we plan to further investigate multiple energy planners with conflicting interests but also to investigate the so-called IMCF-Cloud extensions that will enable IMCF to operate as a CMC controller in the cloud. We also aim to look at CO2 reductions methods with algorithms geared towards the environment. Finally, we aim to investigate power workload identification methods for power-hungry devices (e.g., white devices, electric vehicles, heating) and how to reschedule those workloads in a environmental friendly manner.

REFERENCES

[1] Marileni Angelidou, Constantinos Costa, Artyom Nikitin, and Demetrios Zeinalipour-Yazti. 2018. FMS: Managing Crowdsourced Indoor Signals with the Fingerprint Management Studio. In *19th IEEE International Conference on Mobile Data Management*. pp. 288–289. https://doi.org/10.1109/MDM.2018.00054

[2] Soteris Constantinou, Andreas Konstantinidis, Demetrios Zeinalipour-Yazti, and Panos K. Chrysanthis. 2021. The IoT Meta-Control Firewall. In *37th IEEE International Conference on Data Engineering*. IEEE Computer Society, April 19 - April 22, Chania, Crete, Greece, 12 pages (accepted).

[3] X. Meng, W. Cong, H. Liang, and J. Li. 2018. Design and implementation of Apple Orchard Monitoring System based on wireless sensor network. In *IEEE International Conference on Mechatronics and Automation*. 200–204. https://doi.org/10.1109/ICMA.2018.8484350

[4] Xianghang Mi, Feng Qian, Ying Zhang, and XiaoFeng Wang. 2017. An Empirical Characterization of IFTTT: Ecosystem, Usage, and Performance. In *Internet Measurement Conference*. pp. 398–404.

[5] Steven Ovadia. 2014. Automate the Internet With "If This Then That" (IFTTT). *Behavioral & Social Sciences Librarian* 33, 4 (2014), 208–211. https://doi.org/10.1080/01639269.2014.964593 arXiv:https://doi.org/10.1080/01639269.2014.964593

[6] Daniel Petrov, Rakan Alseghayer, Panos K. Chrysanthis, and Daniel Mosse. 2019. Smart Room-by-Room HVAC Scheduling for Residential Savings and Comfort. In *The 10th Intl. Green and Sustainable Computing Conf.* pp. 1–7.

[7] Blase Ur, Melwyn Pak Yong Ho, Stephen Brawner, Jiyun Lee, Sarah Mennicken, Noah Picard, Diane Schulze, and Michael L. Littman. 2016. Trigger-Action Programming in the Wild: An Analysis of 200,000 IFTTT Recipes. *In ACM CHI Conference on Human Factors in Computing Systems* (2016), 3227–3231.

[8] Lina Yao, Quan Z. Sheng, and Schahram Dustdar. 2015. Web-Based Management of the Internet of Things. *IEEE Internet Computing* 19, 4 (2015), pp. 60–67.