Internet of Things 24 (2023) 100908

Contents lists available at ScienceDirect

Internet of Things

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

Research article

When edge intelligence meets cognitive buildings: The COGITO platform

Marica Amadeo[a], Franco Cicirelli[b], Antonio Guerrieri[b,*], Giuseppe Ruggeri[a], Giandomenico Spezzano[b], Andrea Vinci[b]

[a] University Mediterranea of Reggio Calabria, Via Graziella, Loc. Feo di Vito, Reggio Calabria, 89100, Italy
[b] Institute for high-performance computing and networking of the National Research Council of Italy (ICAR-CNR), Via P. Bucci, Cubo 8/9C, Rende, 87036, Italy

ARTICLE INFO

Keywords:
Cognitive building
Edge intelligence
Computing continuum
In-network processing
Internet of things

ABSTRACT

Future buildings are complex systems that aim at improving the quality of life of their inhabitants and increasing safeness, security, and efficiency. In order to reach these goals, they require their own self-management and self-adaptation capabilities, thus becoming cognitive entities. However, developing cognitive buildings that exploit advanced Artificial Intelligence (AI) techniques in a distributed fashion is still a challenge. Indeed, they need to continuously collect and process a variety of environmental parameters, learn and predict the users' needs and preferences, and then control a large number of heterogeneous devices. These operations may leverage the dynamic availability of edge and cloud computing resources.

This paper discusses the challenges and requirements tied to the development of cognitive buildings and proposes COGITO, an Internet of Things platform specifically suited to design and implement cognitive buildings. COGITO enables edge intelligence and the exploitation of all the available resources in the *computing continuum* between the edge and the cloud. All of this is to support holistic services that guarantee energy efficiency, security, and user comfort. For validation purposes, a case study involving the use of COGITO in an office building is presented, together with a performance evaluation analysis.

1. Introduction

Thanks to the recent advances in the Internet of Things (IoT), modern buildings are becoming complex entities that offer an advanced living experience to their inhabitants [1,2]. By leveraging a variety of interconnected systems, like smart household appliances, microgrids, heating, ventilation, and airconditioning (HVAC) equipment, they are engineered to provide comfort, security, and safety services, as well as to reduce energy consumption and operational costs.

At the same time, smart building management is getting increasingly difficult and burdensome. On the one hand, smart buildings are *information-intensive and dynamic structures*, in which a huge amount of data is constantly generated and processed to enable context-aware services according to the varying needs and preferences of their occupants and the conditions of the environment, e.g., weather and energy costs. On the other hand, smart buildings are *heterogeneous and extremely dense network environments*, where different devices, ranging from resource-constrained sensors and actuators to powerful smartphones and servers, communicate with each other to consume and produce information and accomplish collaborative tasks.

* Corresponding author.
E-mail address: antonio.guerrieri@icar.cnr.it (A. Guerrieri).

https://doi.org/10.1016/j.iot.2023.100908
Received 21 February 2023; Received in revised form 31 July 2023; Accepted 10 August 2023
Available online 22 August 2023

To develop efficient self-management capabilities, today's buildings are evolving towards *cognitive* entities that self-learn, self-organize, and self-adapt [3,4]. More specifically, a cognitive building leverages Artificial Intelligence (AI) to collect and analyze all the data generated by its inhabitants and devices distributed in the environment and, accordingly, it modifies its control functions and services, with human intervention kept at a minimum [5].

AI applications usually require high storage and processing resources on a scale typical of large cloud-based data centers. Although most of today's commercial smart environments rely on cloud computing [6], such an approach poorly matches the requirements of latency-critical applications and affects the core network with very-high traffic volume. Therefore, edge computing has been devised to tackle such challenges by enabling caching and processing services directly at the network edge [7], close to where data is produced and consumed. Resource availability in the proximity of users also tempers some of the privacy risks arising from data storage in remote data centers handled by third parties [8].

The basic idea is to leverage edge and cloud in a complementary fashion [9–11] so as to not only support AI in the cloud but also to distribute AI among edge nodes, thus implementing the so-called edge intelligence paradigm [12,13].

According to recent initiatives like the IETF Computing in the Network (COIN) Research Group [14] and related work [11,15], edge computing nodes are not just purpose-built servers deployed close to the end devices, but also in-network nodes along the edge/cloud continuum [16], like routers and access points, that provide processing and caching capabilities, in addition to traditional communication services. Such in-network devices are characterized by heterogeneous hardware components and highly dynamic availability of their (limited) resources, e.g., their available memory and CPU may change over time depending on the consumer applications' demands. Therefore, a cognitive building platform should deploy specific strategies for dynamically managing and allocating computing tasks among the distributed edge devices. Moreover, it should provide rich programming abstractions to access heterogeneous hardware and to guarantee the interoperability and extendability of its services [17].

In this paper, we show how these targets are tackled by COGITO (a COGnItive dynamic sysTem to allOw buildings to learn and adapt), a novel IoT-based edge/cloud computing platform that leverages lightweight device virtualization and the agent metaphor [18] to bring out the full potential of the cognitive building paradigm. COGITO takes advantage of the available *caching*, *computing*, and *communication* (3C) resources [19] along the edge/cloud continuum to provide cognitive services closer to the end devices, in order to limit the traffic load and meet the requirements of latency-sensitive applications. Unlike previous works, where service orchestration and management are performed in a centralized fashion [3] or involve purpose-built edge servers only [9], COGITO dynamically builds, runs, and manages cognitive services in a distributed fashion, according to an allocation strategy that takes into account the available 3C resources.

More specifically, the paper provides the following main contributions.

- It presents a holistic vision of the cognitive building paradigm by considering the main services involved and their issues.
- It presents the COGITO platform by discussing its main functionalities and management modules. It also presents a computing task allocation strategy deployed cooperatively along the edge/cloud continuum and aimed at minimizing the task execution latency.
- To show the effectiveness of COGITO, the paper presents the real case study of a cognitive office running in the ICAR-CNR headquarters in Rende (Italy). Moreover, it shows a simulation campaign in a large-scale network scenario to assess the performance of COGITO against two traditional cloud-based and edge-based benchmarks. Results show that, by leveraging the pervasive 3C resources along the continuum, COGITO is able to minimize the task execution latency.

The rest of the paper is organized as follows. Section 2 presents the cognitive building paradigm and the related work. The COGITO platform and the deployed cooperative task allocation strategy are introduced in Section 3. The validation of the platform is reported in Section 4. Finally, Section 5 concludes the work.

2. Cognitive buildings: Services and related work

Future buildings are systems of systems integrating things and human beings in the same environments. Human beings spend most of their time in buildings for work, amusements, socialization, and house-living, and have their needs, routines, and preferences, in terms of comfort, security, and costs.

On the other hand, the building itself has its needs, too [20]. For example, the building infrastructures, like the water and energy delivery systems [21,22], need to be maintained efficient and effective [23]; the building's internal and surrounding areas need to be secure for its inhabitants [24]; building spaces need to be used in an efficient and effective manner. In parallel, the building requires to be capable of fast responding to emergencies like fires or quakes [25], in order to preserve the life of the people, and, if possible, its structure [26].

While each of the above needs is usually taken into account one by one, it is clear that many advantages can be reached by addressing these issues altogether in a holistic fashion by implementing some dedicated cognitive services [27].

Besides, given the same physical environment in a building, many different activities can be carried out. For instance, the same room in an office can be used either for a business meeting or a work party, as well as a room in an apartment can be used for fitness or relaxing while reading a book. The environment should learn the specific situation and automatically adapt to the specific activity. As a consequence, a cognitive building needs to continuously: (i) *monitor* its state and that of its inhabitants by continually gathering and processing information; (ii) *learn* about itself, its inhabitants' behaviors, preferences and needs, and the building context as a whole; (iii) *act* and *adapt* in order to make itself comfortable, safe, secure, and efficient. To highlight common functionalities and issues in the realization of a cognitive building, a discussion about a significant set of services [24], that a building should offer, follows. After this, the section will discuss some related work.

2.1. Services of cognitive buildings

In the last few years, research on cognitive buildings focused on various services with heterogeneous targets, ranging from comfort support to anomaly detection. In the following, we describe the most popular ones.

Comfort Management. Different kinds of comfort are required to be managed in a building, namely, climatic, visual, and air quality comforts. The goal is that of ensuring an appropriate level of temperature, humidity, lighting, and air quality in a given environment (room/office/apartment) by considering: (i) the learned profiles of its occupants (e.g., in terms of human presence and activities both forecasted and sensed), (ii) the physical characteristics of the environment (e.g., thermal capacity, sun exposition), (iii) the current regulations, and (iv) energy-saving constrains [28]. The building should automatically adapt to users' habits, changes in inhabitants, building structure, and energy costs through self-learned strategies. This involves controlling HVAC, lighting systems, automated curtains, windows, and doors. Coordination among comfort strategies is essential, as, e.g., automated curtains' opening on sunny days affects both lighting comfort and room temperature.

Appliance Scheduling. Nowadays, people cannot live without many essential and useful appliances because they meaningfully increase their quality of life. Whereas for some of them, the use cannot be delayed (e.g., a hairdryer or an oven), for others there is the possibility to defer/anticipate their use (e.g., washing machine or dishwasher). A cognitive building relieves users in planning the schedule of appliances while taking into account complex constraints such as (i) keeping the total instantaneous power consumption under a threshold, (ii) minimizing the total energy cost by using appliances when electricity cost is lower or when self-produced energy is available, and (iii) satisfying some priorities expressed by the user or learned by observing the normal use-patterns of the appliances themselves [29,30].

Commonplaces Management. Cognitive buildings can help in scheduling, regulating, and maintaining indoor/outdoor common spaces and utilities (e.g., meeting rooms, gardens, and shared laundries). Buildings can predict or know in advance, due to previous bookings, when people are starting something somewhere and act accordingly to prepare the environments and to maintain them after their use (e.g., a party in the garden, talks in a meeting room). A common problem that deserves particular attention is the management of the parking lots [31]. In this case, besides booking, it is very important to regulate the access and the traffic inside the parking.

Anomalies and Dangerous Situations Management. An anomaly is a particular condition or situation that can be categorized as *something strange* or *not occurring usually*. For example, an anomaly may occur when several windows are opened despite windy weather or when a car blocks another one in the parking lot. A cognitive building should be able to self-detect anomalies, issue warnings if necessary, and automatically take appropriate actions to manage the situation [32]. Additionally, it should learn user preferences and habits, as a situation may be considered an anomaly for one user but normal for another. Certain anomalies can be dangerous situations where the building needs to react swiftly to ensure people's safety. For instance, during fires, the building might close gas valves, facilitate evacuations by indicating secure paths, and unlock closed doors/gates to assist rescuers.

Predictive Maintenance. Wrong maintenance for both appliances and building infrastructures may lead to performance degradation or failures with consequent interruption of services and expenses in buying new equipment or for costly repairs [33]. Thus, a cognitive building needs to be able to monitor itself and its appliances and predict in advance when there is a need for maintenance [20] or when the end of the useful life of components is approaching. This service can avoid appliance crashes or breakdowns, by suggesting when and what component needs to be repaired/replaced.

2.2. Related work and contributions

A variety of cognitive building solutions have been designed over the years [34,35]. However, the majority of them are tailored to a single specific service, including HVAC control [36,37], microgrid energy management [38], appliance scheduling [39], parking lot management [31], anomaly detection [40].

To the best of our knowledge, very few works so far have targeted the implementation of a holistic cognitive building system.

In [27], the authors present OCTOPUS, a framework that employs deep reinforcement learning (DRL) to optimally control multiple building's services, including HVAC, lighting, blind, and window management. The target is to simultaneously meet thermal comfort, visual comfort, and indoor air quality goals, while optimizing energy usage. Therefore, a reward function based on energy and comfort is deployed in a centralized control layer, thus raising scalability issues in the presence of large buildings. The framework is trained based on 10-year weather data from two cities with very distinct characteristics and is evaluated through a calibrated simulation model of a real building, without a prototype implementation.

In [41], the Multi-Agent Reinforcement learning COntrol for HVACs (MARCO) framework is proposed, with the target of minimizing HVAC energy while maintaining user comfort. To achieve scalability, MARCO is designed as a multi-agent system with transfer learning, where individual agents model the interacting HVAC sub-systems. A model trained on one task is used as the starting point for another similar task. The same technique is applied in [42] to deploy a multizone thermal control algorithm (MOCA). Like Octopus, however, MARCO and MOCA are associated with a specific reward function and do not consider flexibility and extendibility as design requirements. Also, they are only evaluated in a simulated environment.

A demonstrator of the cognitive building concept has been developed in the eLUX Laboratory[1] at the University of Brescia (Italy) [43]. The building is equipped with an IoT network that gathers and transmits indoor parameters, like temperature, humidity,

[1] https://elux.unibs.it/

Table 1
Comparison of related work on cognitive services.

Work	Targeted service(s)	Reference technologies	Validation
[31]	Smart parking	ML in the Cloud	Simulations
[36]	Thermal comfort	DRL at the Edge	Simulations
[37]	Thermal comfort	DRL at the Edge	Simulations Test-bed
[38]	Energy management	DRL at the Edge	Simulations
[39]	Appliance scheduling	DRL at the Edge	Simulations
[40]	Anomaly detection	ML at the Edge	Simulations
[27]	Holistic	DRL at the Edge	Simulations
[41]	Energy management and thermal comfort	DRL at the Edge	Simulations
[42]	Thermal comfort	DRL at the Edge	Simulations
[43]	Holistic	Digital Twins at the Edge	Test-bed
COGITO	Holistic	Virtual Objects and AI Agents on the Edge/Cloud continuum	Simulations Test-bed

and illuminance, to a management system. This latter controls the HVAC utilities and the window opening according to the expected comfort levels and the energy consumption profile. The users can provide explicit feedback to the management system about their thermal, acoustic, and visual comfort through a user-friendly dedicated application [44].

Compared to the reviewed works, COGITO is a holistic cognitive system that leverages a distributed design based on edge/cloud continuum to support scalability and multi-objectiveness, as it will be clarified in the following section. A preliminary description of the COGITO platform has been presented in [45], together with a summary of the developed cognitive applications for indoor and outdoor environments, namely thermal comfort, occupancy forecast, smart meeting room, air quality, smart parking, monitoring weather conditions. In this paper, instead, we mainly focus on the pervasive in-network computing functionalities deployed by COGITO to manage the available 3C resources and run AI services. Moreover, COGITO performances are evaluated through real exploitation in an office building and, in parallel, through simulations on a large-scale building.

Table 1 summarizes the main differences between COGITO and related work.

3. The COGITO platform

COGITO is an IoT edge computing platform, developed in Java [18], specifically conceived for the realization of cognitive buildings and, more in general, cognitive environments. In the following, we will discuss the main requirements that lead to the platform design and implementation, and hence the main components and functionalities of COGITO.

3.1. Requirements

The COGITO Platform was designed to comply with the following main requirements.

Advanced cognitive abilities. Human behavior, preferences, and needs, together with environmental variables, dynamically change over time. COGITO needs to exploit automatic learning and self-adaptation to adapt to such modifications properly. Furthermore, it must be able to forecast human behavior, environmental dynamics, and future events that may occur in order to prevent unwanted situations or plan actions to pursue the system's goals.

Heterogeneity and multi-objectiveness. Cognitive buildings include devices produced by different vendors and implementing different protocols and semantics. COGITO should be capable of connecting and exploiting all of them, despite their differences. At the same time, different services can exploit a common set of devices, and need to control them to reach different goals that are often in contrast. For instance, to ensure a good level of visual comfort it should be needed to open the curtain in a given room, but this can impair the climatic comfort and/or result in greater energy consumption. Thus, COGITO should implement multi-objective optimization strategies.

Real-time operations. Many of the services provided within a cognitive building have soft or hard real-time constraints, as they enact continuous control behaviors having effects on building infrastructures, appliances, or devices. Such behaviors may need to get information and make decisions at different rates. Delays in communication or processing operations may bring inconsistencies in the results of the actions that may forbid services to accomplish their tasks correctly. Therefore, COGITO should support reliable and low-latency communications and optimize the execution of processing tasks.

Exploiting computation resources along the edge/cloud continuum. Cognitive services heavily differ in demand for computational resources and in the amount of data they require to exchange, produce, and consume. By leveraging the computation resources along the edge/cloud continuum, COGITO should be able to deploy the services in the best location. For instance, resource-intensive computations could be offloaded to the cloud, while stream-based operations with low communication delay could be allocated at the edge. As a consequence, it is required to carefully take into account issues related to network topology and bandwidth, task-scheduling, and allocation policies aware of task complexity, network congestion, CPU load and capabilities, data-source location, and the volume of data to process.

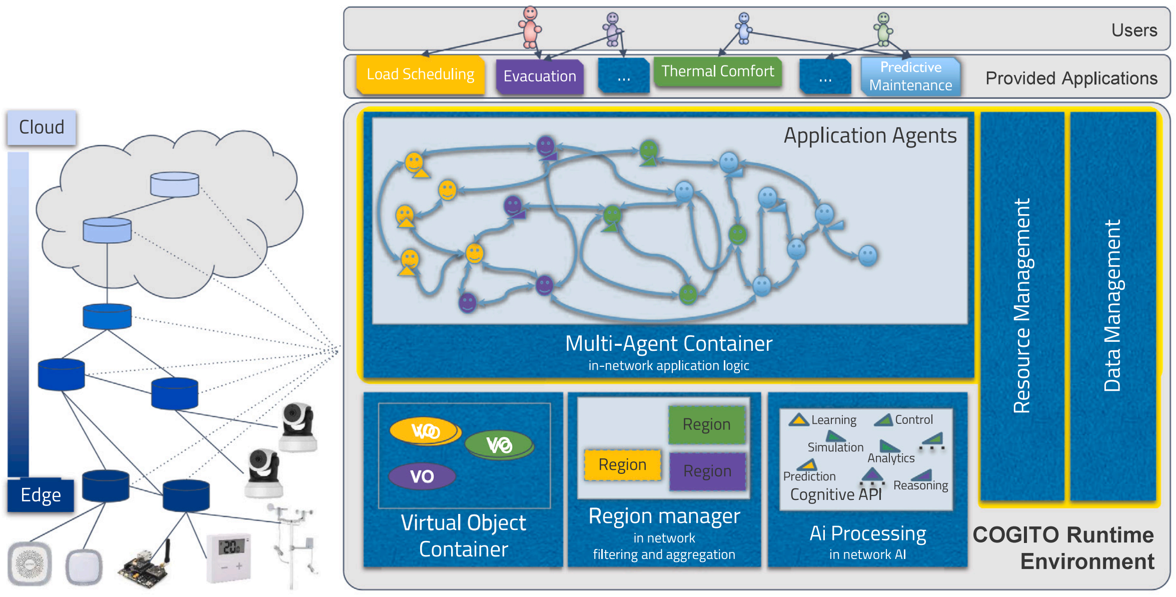


Fig. 1. The COGITO platform and its deployment.

Multiple-scale views. Different services could need different views of the system with different aggregation levels. A service can need an aggregated view of the state of the whole sensors of specific measurements, while another one can need a detailed view of a little set of known devices localized anywhere in the building. Aspects related to information availability, transparency, and time-based requirements need to be taken into account.

Privacy. The devices deployed in the cognitive building, and the provided services, may be perceived by inhabitants as intrusive. For instance, they could question the need to continuously track their activities and movements, or who has access to this information. To be trusted and accepted, COGITO should carefully address privacy issues, by avoiding unnecessary exploitation and storage of personal data by third parties.

Easy maintenance and openness. A cognitive building is a very dynamic and ever-changing environment, requiring the frequent identification and the substitution of broken devices, the continuous management of software updates and the introduction of new devices, sometimes deployed as black box systems by different vendors. Therefore, COGITO should include flexible and friendly APIs for easy maintenance and inclusion of (new) hardware and software components. Moreover, it should be able to interact with open and closed systems, by remaining vendor-neutral.

3.2. Main components

From an infrastructural point of view, the COGITO platform is designed with the capability to exploit, on the basis of users and application needs, the following components (see the left part of Fig. 1):

- a set of physical end-devices (sensors, actuators, and smart objects) that are spread into an environment and are used for monitoring it, producing information, or performing specific actions;
- a cloud infrastructure, which hosts virtual computing nodes suited for high-demanding computing tasks;
- an edge infrastructure, comprising computing nodes spread into the communication network between end-devices and cloud. Such edge nodes can be purpose-built servers or even network nodes enhanced with computing capabilities, e.g., the routers and access points of a campus network. Edge nodes can be organized either in a hierarchical or peer-to-peer topology.

From a software point of view, each computing node hosts a COGITO runtime environment (see the right part of Fig. 1) that supports the continuum computing [46] from the edge to the cloud. Therefore, decentralized, distributed, and cooperative computation is spread over the networked edge/cloud nodes thanks to a homogeneous environment suited to let tasks execute everywhere, independently from the place in which a computing node (edge or cloud) is located. In addition, it provides software abstractions and modules which can be exploited for the design and implementation of cognitive systems, as it will be clarified in the following and summarized in Table 2.

Table 2
Requirements addressed by the COGITO platform.

Requirement	COGITO feature
Advanced cognitive abilities	- Application Agents with in-network logic exploiting AI processing
Heterogeneity and multi-objectiveness	- Virtual Object abstraction - In-network application logic, agents, and AI support complex multi-objective behaviors.
Real-time operations	- Processing at the network edge
Exploiting computation resources along the edge/cloud continuum	- Resource Management module monitoring the local 3C resources and handling task processing - Data Management module monitoring data caching and recording data availability information
Multiple-scale views	- Region Manager for in-network filtering and aggregation
Privacy	- Processing at the network edge - Data Management module handling policies for data access and sharing
Easy maintenance and openness	- COGITO APIs permit to dynamically update and include new software/hardware components through new Agents and VOs - Cognitive APIs permit the exploitation of AI libraries developed in other languages

3.3. In-network application logic

The agent metaphor [47] is exploited by COGITO to realize cognitive environments and support in-network application logic. Agents, by definition, are able to exploit distributed computational resources and operate in a networked context. They naturally have the capability to exhibit intelligent behavior. Moreover, the literature demonstrates the effectiveness of this paradigm for dealing with complex and open systems, even when multi-objectiveness is required [48,49]. Agents enable COGITO to implement parallel, multi-objective, mobile, and scalable applications. The Multi-Agent Container supports agents' lifecycle and communication (see Fig. 1) where, for maintenance and system evolution, agents can be dynamically added and removed.

As a distinguishing feature, agent mobility paired with the COGITO runtime permits to support continuum computing [46] from the edge to the cloud. In this way, processes comprehending both business logic and data elaboration can be dynamically executed everywhere based on both application and user needs. Moreover, agent communication abilities together with the COGITO runtime allows the implementation of both vertical and horizontal interaction patterns among computing nodes. All of this allows distinguishing COGITO with respect to the other edge computing platforms that, to the best of our knowledge, rely on a fixed and hierarchical layered architecture [6].

Two kinds of behaviors for the agents are supported, namely "reactive" and "cognitive". Reactive agents are capable of reacting to events and stimuli from the external environment, making some elaborations, and producing other events/data. Cognitive agents (represented in Fig. 1 with a triangle close to agent icons) implement a *learning cycle* in which an agent learns from past experiences.

As another distinguishing feature, COGITO provides a ready-to-use template named *cognitive controller* in order to deal with the common issues of controlling devices, equipment, and appliances. The template favors and simplifies the development of controllers based on reinforcement learning [50]. Suitable abstractions for dealing with environmental sensing, feedback, rewards, actions, system state and history, and for applying a specific reinforcement learning algorithm are provided.

3.4. Virtual objects

To hide the heterogeneity of end-devices and their protocols, COGITO offers the Virtual Object (VO) abstraction, which is a uniform high-level description of the end-devices' features (e.g., sensed parameters, actuation functionalities) and a collection of standard methods made available to the application agents which monitor and control them. Essentially, a VO exposes an abstract representation (i.e., machine-readable description) of the features and capabilities of physical objects spread in the environment. A change in the physical devices (e.g., for maintenance) and in the adopted communication protocols will only affect virtual objects. The VOs support both asynchronous and synchronous data reading. The asynchronous method relies on an implementation of the publish/subscribe pattern. A VO is dynamically deployed on the computing node from which the physical objects to manage are accessible. A VO is usable by the software agents located on the same computing node where it is deployed. The *Virtual Object Container* (see Fig. 1) manages the VOs lifecycle.

3.5. In-network data filtering and aggregation

A common research issue for IoT platforms [6,17] is to make available mechanisms and abstractions useful to (i) share, gather, filter, aggregate, and cluster data coming from streams of volatile data; and (ii) natively support complex communication patterns which go beyond the exchange of basic messages.

As a novelty, COGITO introduces the concept of *Region* to support the above operations along with multiple-scale dynamic views on distributed data streams. Regions consider a time-sliding window model and are defined on a subset of the states of devices and agents in the system. A region provides snapshots of the considered states that are made available to client agents by following both a time-based and an event-based approach, e.g., based on the occurrence of specific conditions defined in the data. For optimization reasons, each snapshot is computed once and shared with all the agents requesting it.

Computed snapshots are furnished transparently with respect to the identity and location of data producers. On receiving a snapshot, an agent can use it to transparently contact the involved producers so as to propagate data to them. In such a way, Region implements a recipient-unaware one-to-many communication infrastructure within the platform.

Regions' life-cycle, snapshot caching, and sharing policies are managed by the Region Manager component.

3.6. In-network AI processing

Agents that require Artificial Intelligence and Machine Learning computation can exploit the *AI processing* module. This module allows defining agents' behavior by using third-party libraries and/or external frameworks exploitable for, e.g., forecasting, anomaly detection, user's preferences understanding, planning, unsupervised and semi-supervised learning on data streams, time-series analysis, deep learning, and reinforcement learning [51,52].

The AI processing module can integrate external libraries and frameworks developed in different programming languages, through the so-called *Cognitive APIs*. Such APIs, which represent another novelty of COGITO, specify a standardized way for weaving agents' behavior with external software computational resources, dealing with data exchange, concurrency management, and interoperability issues. For instance, Cognitive APIs were used for exploiting the integration of COGITO with the TensorFlow platform for machine learning.

The cognitive APIs provide mechanisms for supporting the life cycle of a cognitive model (e.g., a neural network) in terms of model deployment, training, and execution. Moreover, such APIs allow to distribute the processing flow among multiple nodes, both in the cloud and in the edge. For instance, a model can be firstly deployed and trained in a cloud node and, subsequently, deployed and executed on multiple nodes on the edge.

3.7. Management modules

Existing platforms for cognitive (and smart) buildings, like the ones in [3,4,9], leverage purpose-built edge nodes, like servers or gateways, to perform a variety of computing services. Although these latter could be containerized and deployed in different locations, no specific strategies are defined for on-the-fly service allocation. Vice versa, COGITO potentially runs in any network node enhanced with computing capabilities and it provides specific modules that handle the local 3C resources and dynamically (re)allocate services (and their consequent execution) in a way that is collaborative and transparent to the applications. The conceived Data and Resource Management modules work in coordination with their counterparts on the other COGITO nodes to achieve better usage of the 3C resources distributed between the edge and the cloud.

More specifically, the *Data Management* handles data caching, records data availability information (e.g., forwarding rules, latency), and manages the policies for data access and sharing (e.g., privacy and priority). Since storage resources are limited, the caching scheme selects what data to cache and applies a replacement operation when the cache is full or the data is stale. The caching decision applies to raw and processed data, and it is based on a popularity metric: the highly requested contents are cached by the edge nodes in order to limit the retrieval latency and the traffic load. The storage can be demanded to the cloud in the presence of low-popular contents or when long-term analysis must be applied.

The *Resource Management*, instead, monitors the local 3C resources and handles the procedures for agent allocation, Region computation, and AI processing. Such operations, referred to as tasks, can be allocated and moved through the computation nodes spread in the edge/cloud network.

Without loss of generality, in the current implementation, COGITO admits a cooperative allocation scheme called *Best on Path* (BoP) that considers the available 3C resources to select as executor the node, in the path between the requester (i.e., an access node connected to certain end-devices) and the cloud, that minimizes the task execution latency. However, other allocation strategies can be introduced that target other objectives.

3.8. Best on path strategy

In BoP, a potential executor node n_i first checks if the requested task is available in the local cache, e.g., because it has already been computed. If so, it immediately provides the result to the requester. Otherwise, it estimates the execution latency as the sum of the three following terms.

- The input communication latency, which is the time needed to collect the data to be processed. This contribution is negligible if the data are already available locally. Otherwise, it can be estimated by considering the size of the input data and the distance between the potential executor n_i and the data source, e.g., expressed in terms of round-trip-time (RTT) available from the routing table.
- The computing latency, which is computed as the sum of two contributions: (i) the local queuing time, and (ii) the time needed to actually process the data. The first contribution is the waiting time of the task, due to possible previous tasks in the processing queue of n_i. The second contribution depends on the processing resources available at the node, expressed in terms of CPU cycles, and the processing demands of the task, also expressed in the number of CPU cycles.

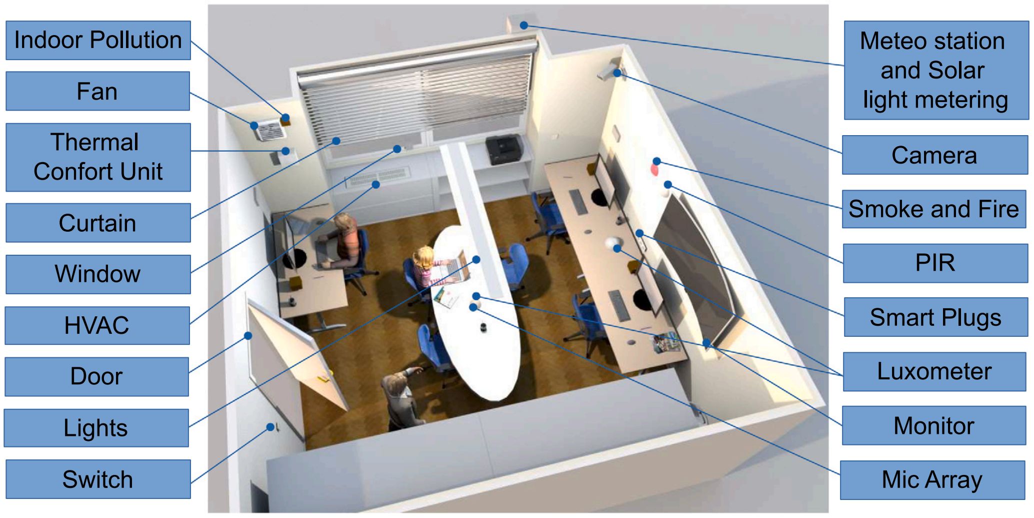


Fig. 2. A rendering of the IoT Laboratory at ICAR-CNR, Rende, Italy. All the highlighted equipment refers to smart sensors, actuators, and controllable devices.

- The output communication latency, which is the time needed to send back the computed result to the requesting node. It depends on the network distance between this latter and the executor and on the size of the output.

When the first edge node, i.e., an access node n_a, is requested to perform an in-network computation, it first calculates the local *execution cost*, as the sum of the three latency contributions. Then, n_a starts a signaling procedure to discover if the cloud or an on-path edge node is able to provide the same task in a lower time. More specifically, n_a transmits towards the cloud a discovery message carrying its own execution cost, and each on-path edge node overwrites the current information if it is able to guarantee a lower cost. At the reception of the message, the cloud starts the task execution if its own execution cost is the lowest. Otherwise, it sends a confirmation message to the node with the lowest cost that will be in charge of the processing.

4. Platform validation

The goal of this section is to provide evidence of the effectiveness of the proposed COGITO platform. More in particular, we will first present a real case study implemented and running in the ICAR-CNR headquarter in Rende (Italy). This case study regards the realization of a cognitive office and is devoted to highlighting the COGITO capabilities in the development of cognitive systems. Then, we will show a performance evaluation study proving how the exploitation of COGITO mechanisms permits achieving good execution performances by leveraging computing in the cloud/edge continuum.

4.1. A COGITO cognitive office

The prototype of a cognitive office has been realized in the context of the IoT Laboratory at the ICAR-CNR headquarters located in Rende (Italy). The system, named CogLab, belongs to a broader IoT infrastructure which covers the whole ICAR-CNR building [24]. The infrastructure aims at realizing a cognitive building concept by hosting IoT/AI applications related to security, energy efficiency, and comfort.

The IoT Laboratory, usually exploited both for meetings and research activities, is a workspace designed for testing and developing IoT applications effectively. It is equipped with several sensors and actuators connected and operating all together through the use of COGITO. The laboratory is also provided with different kind of computing nodes hosting COGITO, e.g., Raspberry Pi 4, Nvidia Jetson Nano, Google Coral, and workstations, that are available for developing cognitive applications. The computing nodes, sensors, actuators and controllable devices are interconnected by using a hierarchically arranged network infrastructure capable of supporting multiple protocols, e.g., Zigbee, Wifi, Ethernet, and DALI.[2] A rendering of the laboratory, along with a list of the equipped smart devices, is depicted in Fig. 2.

The CogLab is a cognitive application devoted to managing the lighting comfort of people inside. Its goal is to recognize the ongoing activities and users' preferences for properly and adaptively acting upon the controllable curtains and the artificial lights. To pursue such a goal, the CogLab exploits the following smart devices, available in the IoT Laboratory: an Ethernet *SV3C-D05* camera, a controllable curtain made smart through a WiFi Shelly 2.5PM, a set of Dali lights, a set of ZigBee Xiaomi Light Detection

[2] Digital Addressable Lighting Interface, https://www.dali-alliance.org/dali/.

Table 3
Exploitation of the used COGITO features.

Functional requirements	Challenges	Used COGITO features	Discussion
People detection. CogLab needs to estimate the position of the people inside the laboratory. This is achieved by exploiting TensorFlow Python library by using a pre-trained model on the video stream gathered by the camera. The video stream is a privacy sensitive data.	- Use of a cross-language software library - Management of privacy-sensitive data - Device abstraction	- Cognitive API - Edge computing - Virtual Object	The Cognitive API are used to integrate the Python and Java environments. All the data are processed locally at the edge nodes thus preserving privacy. A camera virtual object hides details of device management.
Activity Recognition. Based on the position of the people inside the laboratory, CogLab estimates the ongoing activity which is a privacy sensitive information.	- Management of privacy-sensitive data	- Edge Computing	All the data are processed locally thus preserving privacy.
Luminance Monitor. CogLab needs to know light intensity in different zones of the laboratory either as mean value or highest (dazzle) value.	- In-network data filtering and aggregation	- Virtual Object - Region	COGITO Region are used to automatically gather (transparently through virtual object), aggregate and process data coming from light sensor directly in the network.
Luminance Control. CogLab needs to control both the wi-fi curtain and the DALI artificial lights.	- Heterogeneous device management	- Virtual Object	COGITO virtual objects hide both protocol and device heterogeneity.
Light Comfort Management. CogLab exploits an RL-based approach for luminance management.	- Distribution of Reinforcement Learning computation between edge and cloud	- Edge/Cloud continuum - Cognitive API	Cognitive API permits to transparently execute the Reinforcement Learning model training phase over the Cloud and the model execution at the Edge.

Sensor YTC4043GL. Besides the IoT-based control, the curtain and the lights can also be manually operated by the workers. The CogLab is deployed on two Raspberry Pi 4 edge nodes, namely *RP1* and *RP2*, and a Workstation, standing as a cloud node. RP1 edge nodes permits the connection of all the Dali, ZigBee, WiFi, and Ethernet devices by leveraging VOs specifically implemented. The other edge node, instead, runs the components determining the behavior of the CogLab. All the nodes and the workstation run the COGITO runtime.

In order to implement the application logic, the following main agents were developed:

- *PositionDetector*: it analyzes the video stream acquired by the camera, in order to identify the presence, the position, and the number of people inside the room. The recognition of people inside the stream is achieved by using the well-known YOLOv4 model implemented in Python/TensorFlow 2 [53].
- *ActivityRecognition*: it gathers the people's positions and exploits them to recognize a set of possible activities which are currently performed in the room, i.e., *No-activity* (Fig. 3), *Meeting* (Fig. 4), *Deskworking* (Fig. 5), and *Free time* (Fig. 6). This is done by exploiting a predefined set of logic rules.
- *CurtainManager*: by taking into account how people manually interact with the curtain, the sun's inclination and enlightenment, and the current activity inside the room, this cognitive agent learns how to manage the state of the curtain (angle and opening) to optimize comfort and avoid dazzles.
- *LightingManager*: by taking into account how people manually interact with the lighting system, the indoor luminosity, and the current activity in the room, this cognitive agent learns how to manage the artificial lighting system.

The CurtainManager and the LightingManager are developed by exploiting the approach, based on Reinforcement Learning, presented in [54]. In this case study, such an approach is customized for each different detected activity and developed on COGITO.

Some screenshots related to the running CogLab are reported in Figs. 3–6. The screenshots refer to the output of the PositionDetector agent which, fed to the ActivityRecognition agent, allows the identification of the ongoing activity in the IoT Laboratory. This permits the managers to properly operate on the controllable devices in the room so as to ensure a good level of light comfort.

In particular, Fig. 3 shows the case in which no activity is performed in the IoT Laboratory and, as a consequence, all the lights are turned off, and the curtain is closed. In Fig. 4, a meeting is detected as all the people in the room are located around the central table. In such a scenario, the lights are on, and the curtain is fully opened. Fig. 5 represents a scene in which all the people are working on their own workstations. In the shown case, the natural light from the window is enough to reach a good comfort level for the working activity. As a consequence, the artificial lights are off, and the curtain is half-opened. Finally, in Fig. 6, free-time activities are detected. In this last case, the lights are dimmed to 50%, and the curtain is fully opened to maximize the natural lighting in the room.

The benefits of using COGITO are discussed in the following. More in particular, Table 3 highlights the advantages related to the implementation of system functionalities. The first column of the table describes the functional requirements. The challenges in implementing such requirements are reported in the second column, while the third one highlights the used COGITO feature whose corresponding exploitation is described in the last column. By summarizing, the Cognitive API allows the direct exploitation of cross-language machine learning libraries by permitting to choose the most suitable one for a given scenario. All the environmental information needed for monitoring and management purposes is aggregated, filtered, and directly made available to the application agents by the Region module, thus significantly simplifying agent development.

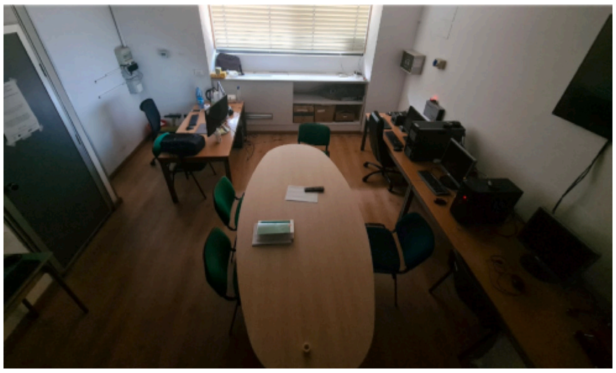


Fig. 3. A screenshot referring to the case in which No-activity is performed in the IoT Laboratory. The artificial lights are off, and the curtain is closed.

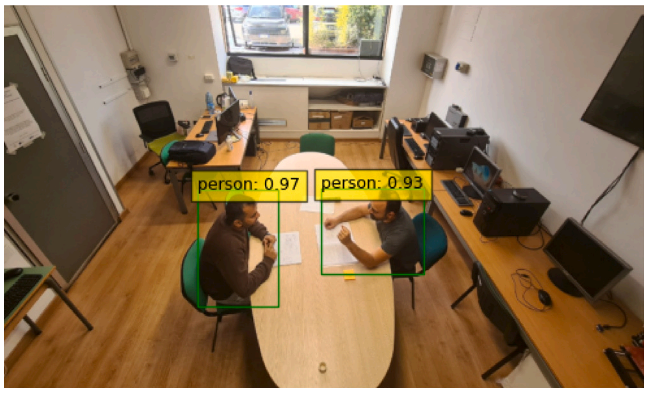


Fig. 4. A screenshot referring to the case in which a meeting activity in which people are discussing around the central table is detected. The artificial lights are on, and the curtain is fully opened.

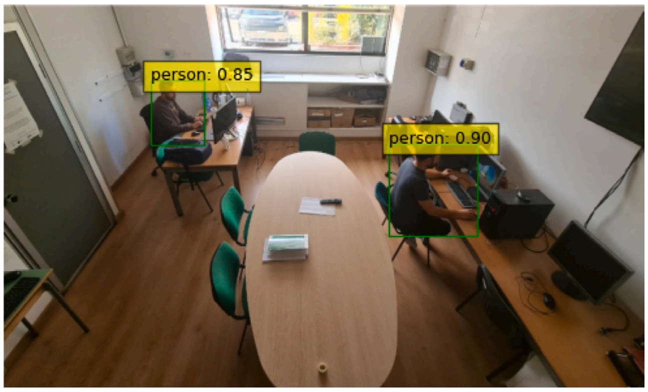


Fig. 5. A screenshot referring to the case in which people in the IoT Laboratory are engaged in Deskworking activities. The artificial lights are off, and the curtain is half-opened.

Besides functional requirements, COGITO is also effective in dealing with non-functional aspects. More in particular, the Resource Management modules transparently permit to automatically spread and balance the computation of the PositionDetector, the CurtainManager, and the LightingManager among the cloud and the most suited edge node so as to balance 3C resources thus also improving system response time. The exploitation of edge computing paired with the agent paradigm permits to promote system scalability and extendibility in that the CogLab system can be easily replicated, scaled to a whole building, and integrated with other functionalities. Finally, COGITO fosters maintenance, replacement, and change of devices in a deployed system, since only the related VOs need to be updated, without affecting the other entities in the system.

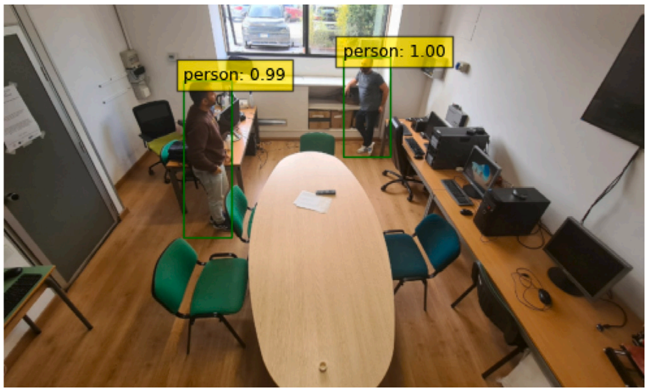


Fig. 6. A screenshot referring to the case in which freetime activities in which people are moving around are detected. The artificial lights are dimmed to 50%, and the curtain is fully opened.

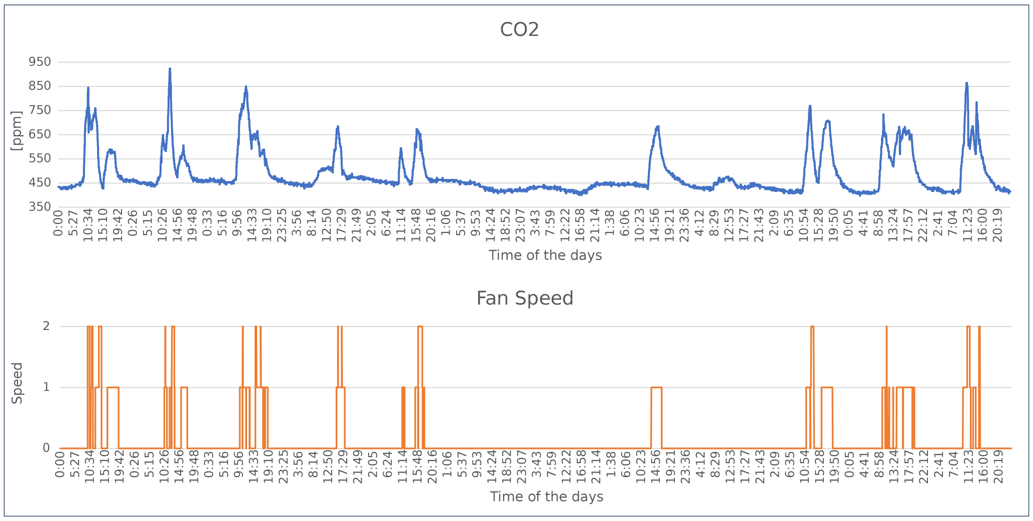


Fig. 7. CO_2 concentration and fan activity in the IoT Laboratory during twelve days.

4.2. *Extending the CogLab functionalities*

In order to prove how the COGITO platform supports the integrations of newly built use cases upon a running system, in the following, we will show how a new functionality can be added to the CogLab. As already shown in Fig. 2, the IoT Laboratory, along with others, is equipped with indoor pollution sensors, which comprehend a CO_2 sensor and a multi-speed controllable fan devoted to forcing air exchange with the external environment. To improve comfort management by also keeping into account the indoor air quality, a further COGITO agent, namely the *AirQualityManager*, has been implemented along with the needed VOs for enabling COGITO to control the fan and gather CO_2 sensor data. The AirQualityManager keeps track of the CO_2 trend and switches the fan on to the appropriate speed when the value read goes over a threshold or its value suddenly increases. The VOs introduced have been installed on the RP1 node, while the manager has been deployed on the RP2 node without stopping the already running CogLab.

To show the effectiveness of the added functionality, Fig. 7 shows the CO_2 concentration in the IoT Laboratory and the fan activity for twelve consecutive days. From the figure emerges that the manager properly controls the fan speed in order to keep the CO_2 concentration in a proper range [55].

4.3. *Evaluation in a four-storey cognitive building*

In this section, we provide a performance evaluation study, carried out by using $Matlab^{®}$, devoted to assessing the execution performance of COGITO when edge/cloud continuum computing is exploited. Such a study wants to reproduce, on a large scale, the use case described above by considering an entire cognitive building.

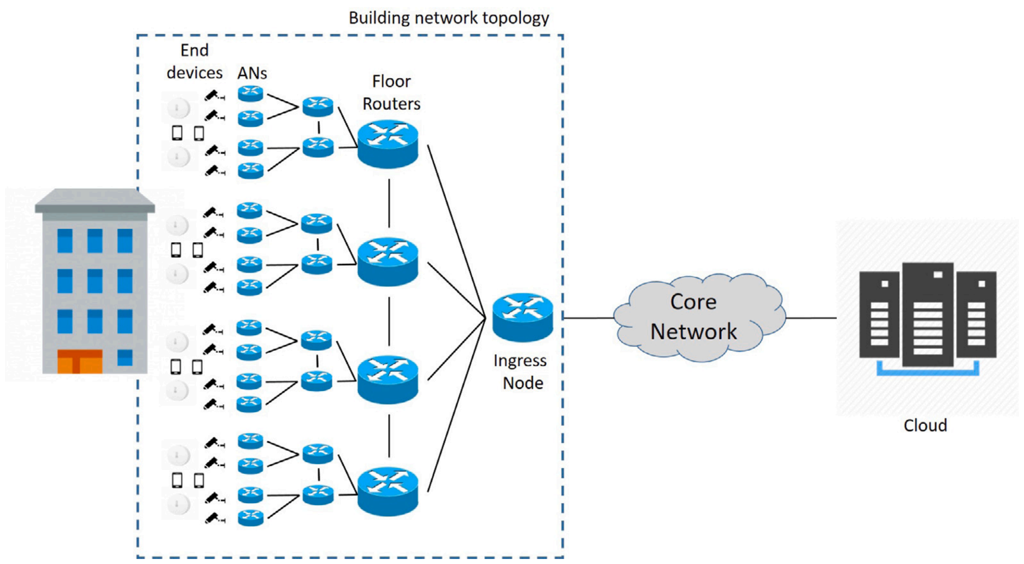


Fig. 8. The simulated network topology.

Table 4
Main simulation settings.

Parameter	Value
Request arrival rate	[100-4300] requests/s
Link latency	Edge [5, 10] ms, Cloud [50] ms
Node computing capability	Edge [20-1500] frame/s, Cloud [20K] frame/s
Average frame size	σ_1 = 1MB (3Mpx), σ_2 = 10MB (15Mpx)

4.3.1. Settings

We consider a four-storey building in which a network of COGITO nodes is used to manage the rooms running the CogLab system. The building network topology, deployed according to the ISO/IEC 11801 standard [56] and shown in Fig. 8, consists of an ingress node that is connected to 4 routers (one per floor) in a meshed topology. Floor routers are, in turn, connected in a three-layered fat-tree topology to leaf nodes. These latter act as access nodes (ANs) to which the end-devices, e.g., cameras, sensors, etc., are connected to. The ingress node links the building network to the core network from which the remote cloud can be reached. For the sake of simplicity, the core network is modeled as a single link with a latency of 50 ms [57,58].

According to the real devices deployed in the considered case study, ANs simulate Raspberry-Pi 4 Model B devices, characterized by a 64-bit quad-core processor and 4 GB SDRAM. Intermediate edge nodes, i.e., parents of ANs and floor routers, consist of Raspberry-Pi 4 Model B devices, augmented with 8 GB SDRAM and Intel Neural Compute Sticks [59] and co-located with Commercial Off-The-Shelf Ethernet switches. The distinct computing resources translate into different processing capabilities of video frames, which have been measured in various experiments and result in the range [20-1500] frames/s.

Each node in the building network topology implements the BoP strategy to allocate tasks cooperatively.

In this evaluation, as a reference computing task, we focus on the processing of video frames because it is the most resource-intensive task of the considered scenario. Video processing is requested by a variable number of clients, e.g., position detection agents, to enable a variety of functions, like people detection and activity recognition.

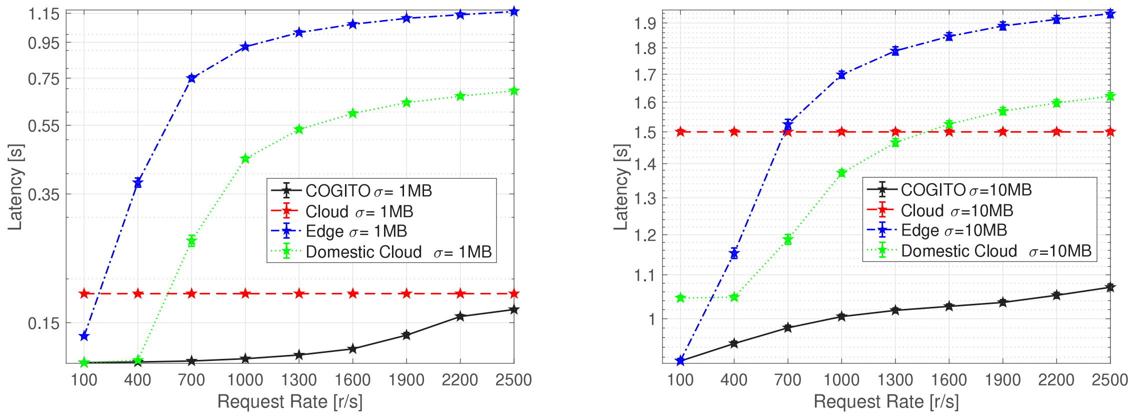
We also assume that the considered video streams have two different video frame dimensions, which are $\sigma = 1$ MB and $\sigma = 10$ MB, as reported in Table 4.

COGITO performance is compared against three benchmark schemes, namely:

- a traditional cloud computing architecture, where a remote data center is in charge of executing all the tasks;
- a distributed edge framework (simply labeled as *Edge* in the following), where a set of edge servers are co-located with the ANs, thus resembling the approach in [3,43];
- a domestic cloud architecture, where a single powerful edge server, co-located with the ingress node, is in charge of executing all the tasks.

We observe that, in the Edge case, edge servers cannot smartly cooperate with each other to share the computation load. If the computing demands exceed the capability of the server, services are queued and executed in order of arrival.

Results are averaged over 100 runs and reported with 95% confidence intervals.



(a) σ = 1MB.　　　　(b) σ = 10MB.

Fig. 9. Average task execution latency when varying the request rate and the frame resolution.

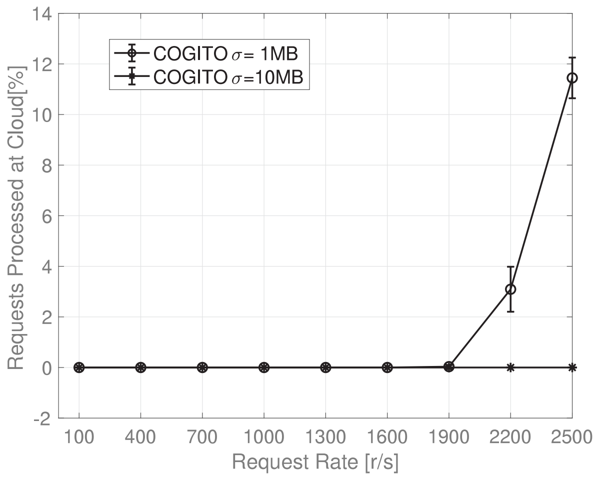


Fig. 10. Fraction of requests processed by the cloud.

4.4. Results

Fig. 9 shows the average task execution latency when considering the frame resolutions σ = 1 MB and σ = 10 MB.

It can be observed that, in both cases, the latency reasonably increases with the request rate for the Edge, Domestic Cloud, and COGITO approaches, due to the increase in the computation load. A constant trend can be seen for the Cloud approach because it can leverage the highest computation resources and, therefore, an increase in the request rate does not affect the overall computation latency.

When the size of the input content is limited, i.e., σ = 1 MB, the communication cost has a low impact on the overall task execution latency, while the computing latency has a key role in the performance. As a result, when the request rate increases, Edge and Domestic Cloud approaches are outperformed by Cloud and COGITO approaches. Moreover, by implementing the BoP cooperative allocation scheme, which allows to effectively leverage the resources along the edge/cloud continuum, COGITO is able to guarantee the lowest latency.

The benefits of the COGITO approach get more remarkable when σ = 10 MB. In this case, the execution latency increases for all the considered approaches, due to the higher communication costs. However, COGITO is able to properly distribute the tasks between the edge nodes and the cloud, thus maintaining the lowest latency.

To get further insights into the COGITO performance, Fig. 10 shows how COGITO distributes the tasks between the edge nodes and the cloud when varying the request rate. It can be observed that, when the frame resolution is high (σ = 10 MB), all the tasks are allocated at the edge, since transferring the data to the cloud would be too costly. Vice versa, when σ = 1 MB, as the request rate increases, a higher fraction of the tasks are allocated in the cloud. This confirms the ability of COGITO to leverage the distributed edge resources to properly allocate the tasks based on their communication and computing costs.

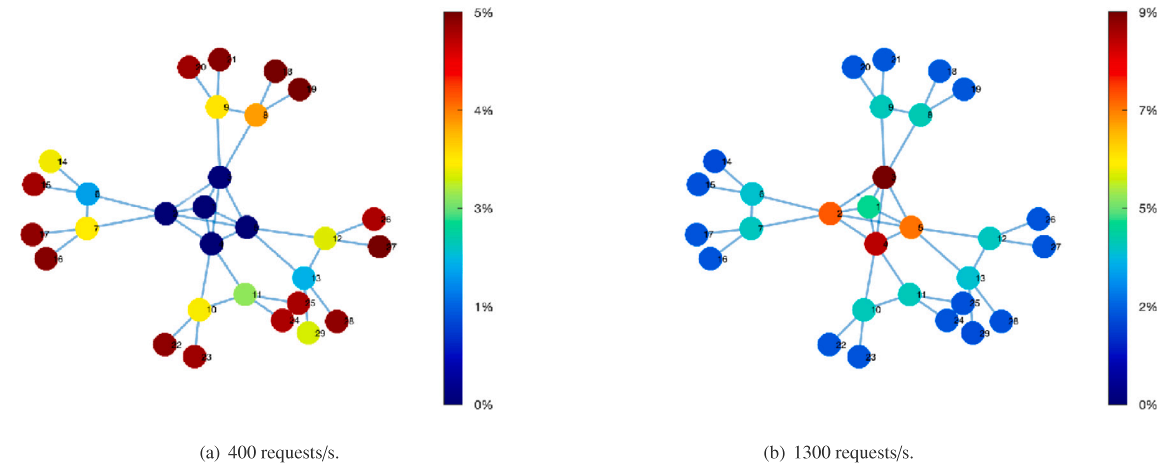


Fig. 11. Workload distribution in the building network topology (percentage of tasks), when varying the rate of requests (σ = 10 MB). The ingress node is labeled as node 1, while the floor nodes are labeled as nodes 2–5, and so on.

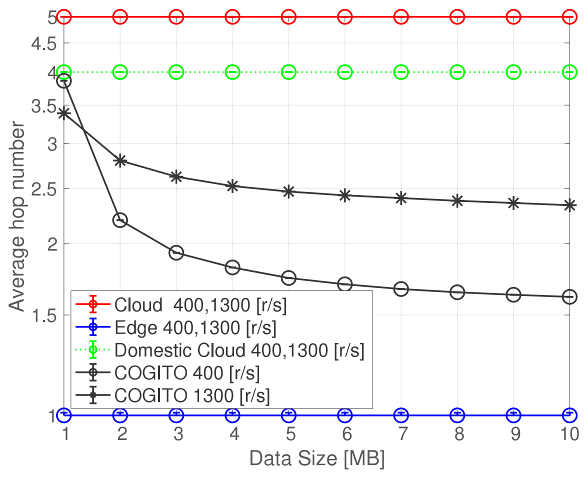


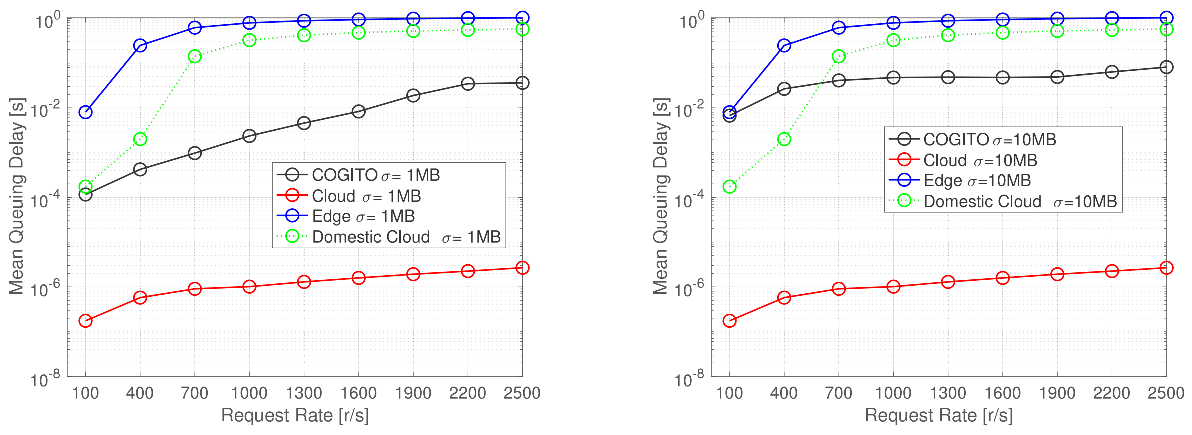
Fig. 12. Average distance, in terms of hop number, between the task executors and the end-devices, i.e., data sources, when varying the request rate [r/s] and the input data size.

To further corroborate this observation, Fig. 11 shows the workload distribution in the building network topology when considering two distinct request rates, namely 400 and 1300 requests/s, and σ = 10 MB. In particular, the heatmap reflects the percentage of tasks allocated in each node and depicts in red the nodes executing the higher percentage of tasks, and in dark blue the nodes that are unloaded.

It can be observed that, in the presence of 400 requests/s, the tasks are mainly allocated to the ANs and their parents, which are shown as red-orange nodes in Fig. 11(a). In particular, the ANs execute about 77% of all the tasks, while the ingress and the floor nodes, which have the higher computing capabilities in our scenario (see Table 4), are not loaded at all. This result is due to the fact that the time needed to collect the input data is a dominant contribution to the overall execution latency calculation. Therefore, the nodes closer to the producers of the video streams can guarantee a lower execution latency. At the same time, however, the ANs cannot handle an increase in the request rate. This is why, when the number of requests/s is 1300, to limit the increase in the latency, the load is distributed across all the edge nodes. In particular, the ANs execute about 33% of the tasks, while the remaining workload is managed by the other nodes.

Fig. 12, instead, shows the average distance, in terms of hop number, between the end-devices, producing the input data to be processed, and the task executors. The metric is computed when considering two distinct request rates, namely 400 and 1300 requests/s (r/s), and when varying the input data size from 1MB to 10MB.

As expected, the hop number does not change for the considered benchmark approaches, since the location of the task executors is fixed. In particular, the hop number is equal to four for the Domestic Cloud approach, since the executor is co-located with the ingress node, and it is equal to five for the Cloud approach since, in the simulated network topology, for the sake of simplicity, the core network separating the edge network from the cloud is modeled as a single link. The lowest hop number is obtained with the



(a) $\sigma = 1MB$. (b) $\sigma = 10MB$.

Fig. 13. Average queuing delay when varying the request rate and the input data size.

Edge approach, where the task executors are co-located with the ANs that are 1-hop far away from the end-devices. This approach limits the traffic in the network, but it is not able to adapt to the varying processing demands of the tasks and has the poorest latency performance when the task request rate increases, as shown in Fig. 9.

It is instead interesting to assess how the hop number changes with the COGITO approach. It can be observed that the distance decreases when the data size increases since the contribution of the *communication latency*, i.e., the time needed to collect the input data, becomes significant, and COGITO has to move the task execution closer to the end-devices. In parallel, it can be noticed that the hop number highly differs with the request rate: when the computing demand is higher (i.e., 1300 requests/s), the average hop number increases since it is necessary to distribute the load among more nodes, including those that are further away from the end-devices. The improved load balancing capability of COGITO is also reflected by another performance metric, namely the average queuing delay, which measures the average queuing time experienced by tasks before their actual execution, i.e., when the computing demands exceed the capability of the executor, services are queued and executed in order of arrival. Fig. 13 shows the queuing delay when varying the input data size (1MB and 10 MB) and the request rate (from 100 to 2500 requests/s). It can be observed that, reasonably, even if the Cloud approach shows the lowest queueing delay, COGITO outperforms all the other edge solutions.

5. Conclusion

This article presented COGITO, a platform suited to develop cognitive buildings by coupling the agent-based metaphor and edge intelligence with in-network computing. As shown by the proposed case study and evaluation, COGITO simplifies and assists the development of cognitive environments and effectively distributes the processing load in the edge/cloud continuum. Additionally, the ability to integrate multiple data sources and use advanced machine learning algorithms enables the creation of sophisticated cognitive systems. By realizing the convergence of communication, caching, computing, control, and sensing, and by pushing cognitive services at the edge, COGITO paves the way to the future 6G applications [60].

In future work, we plan to define other task allocation strategies that, in addition to the minimization of the latency, target other important objectives, like the reduction of the data traffic in the building network, or the load balancing between the COGITO nodes.

Future enhancements of COGITO will also be devoted to:

- the support of a model-driven development approach, based on methodologies such as the ones described in [61,62];
- the integration of the newest IoT protocols (e.g., Matter);
- extending COGITO with modules purposely made for profiling distributed applications and performance monitoring.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: All the authors report financial support was provided by the Government of Italy Ministry of Education University and Research.

Data availability

The data that has been used is confidential.

Acknowledgments

This work has been partially supported by European Union - NextGenerationEU - National Recovery and Resilience Plan (Piano Nazionale di Ripresa e Resilienza, PNRR) - Project: "SoBigData.it - Strengthening the Italian RI for Social Mining and Big Data Analytics" - Prot. IR0000013 - Avviso n. 3264 del 28/12/2021. This work has also been partially supported by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on "Telecommunications of the Future" (PE00000001 - program "RESTART"). This work has also been partially supported by the National Research Council of Italy (CNR) [http://dx.doi.org/10.13039/501100004462], "Le Scienze per le TRansizioni Industriale, Verde ed Energetica": Towards Sustainable Cognitive Buildings (ToSCoB) project.

References

[1] M.S. Aliero, M. Asif, I. Ghani, M.F. Pasha, S.R. Jeong, Systematic review analysis on smart building: Challenges and opportunities, Sustainability 14 (5) (2022) 3009.
[2] D. Minoli, K. Sohraby, B. Occhiogrosso, IoT considerations, requirements, and architectures for smart buildings - Energy optimization and next-generation building management systems, IEEE Internet Things J. 4 (1) (2017) 269–283.
[3] S. Rinaldi, et al., Metrological issues in the integration of heterogeneous lot devices for energy efficiency in cognitive buildings, in: IEEE I2MTC, 2018, pp. 1–6.
[4] J. Ploennigs, A. Ba, M. Barry, Materializing the promises of cognitive IoT: How cognitive buildings are shaping the way, IEEE Internet Things J. 5 (4) (2017) 2367–2374.
[5] M.O. Osifeko, G.P. Hancke, A.M. Abu-Mahfouz, Artificial intelligence techniques for cognitive sensing in future IoT: state-of-the-art, potentials, and challenges, J. Sens. Actuator Netw. 9 (2) (2020) 21.
[6] B. Qolomany, et al., Leveraging machine learning and big data for smart buildings: A comprehensive survey, IEEE Access 7 (2019) 90316–90356.
[7] W. Shi, J. Cao, Q. Zhang, Y. Li, L. Xu, Edge computing: vision and challenges, IEEE Internet Things J. 3 (5) (2016) 637–646.
[8] P. Ranaweera, A.D. Jurcut, M. Liyanage, Survey on multi-access edge computing security and privacy, IEEE Commun. Surv. Tutor. 23 (2) (2021) 1078–1124.
[9] Y. Liu, C. Yang, L. Jiang, S. Xie, Y. Zhang, Intelligent edge computing for IoT-based energy management in smart cities, IEEE Netw. 33 (2) (2019) 111–117.
[10] F.-J. Ferrández-Pastor, H. Mora, A. Jimeno-Morenilla, B. Volckaert, Deployment of IoT edge and fog computing technologies to develop smart building services, Sustainability 10 (11) (2018) 3832.
[11] Y. Hao, Y. Miao, L. Hu, M.S. Hossain, G. Muhammad, S.U. Amin, Smart-Edge-CoCaCo: AI-enabled smart edge with joint computation, caching, and communication in heterogeneous IoT, IEEE Netw. 33 (2) (2019) 58–64.
[12] V. Barbuto, C. Savaglio, M. Chen, G. Fortino, Disclosing edge intelligence: A systematic meta-survey, Big Data Cogn. Comput. 7 (1) (2023) 44.
[13] S. Deng, H. Zhao, W. Fang, J. Yin, S. Dustdar, A.Y. Zomaya, Edge intelligence: The confluence of edge computing and artificial intelligence, IEEE Internet Things J. 7 (8) (2020) 7457–7469.
[14] M. McBride, et al., Edge Data Discovery for COIN, IETF Computing in the Network Research Group, 2020.
[15] C. Cicconetti, M. Conti, A. Passarella, In-network computing with function as a service at the edge, Computer 55 (9) (2022) 65–73.
[16] L. Baresi, D.F. Mendonça, M. Garriga, S. Guinea, G. Quattrocchi, A unified model for the mobile-edge-cloud continuum, ACM Trans. Internet Technol. (TOIT) 19 (2) (2019) 1–21.
[17] R. Morabito, V. Cozzolino, A.Y. Ding, N. Beijar, J. Ott, Consolidate IoT edge computing with lightweight virtualization, IEEE Netw. 32 (1) (2018) 102–111.
[18] R.C. Cardoso, A. Ferrando, A review of agent-based programming for multi-agent systems, Computers 10 (2) (2021) 16.
[19] E.K. Markakis, K. Karras, A. Sideris, G. Alexiou, E. Pallis, Computing, caching, and communication at the edge: The cornerstone for building a versatile 5G ecosystem, IEEE Commun. Mag. 55 (11) (2017) 152–157.
[20] S. Agostinelli, Cognibuild: Cognitive digital twin framework for advanced building management and predictive maintenance, in: E. Arbizzani, E. Cangelli, C. Clemente, F. Cumo, F. Giofrè, A.M. Giovenale, M. Palme, S. Paris (Eds.), Technological Imagination in the Green and Digital Transition, Springer International Publishing, Cham, 2023, pp. 69–78.
[21] G. Garofalo, A. Giordano, P. Piro, G. Spezzano, A. Vinci, A distributed real-time approach for mitigating CSO and flooding in urban drainage systems, J. Netw. Comput. Appl. 78 (2017) 30–42, http://dx.doi.org/10.1016/j.jnca.2016.11.004.
[22] S.S. Reka, T. Dragicevic, Future effectual role of energy delivery: A comprehensive review of Internet of Things and smart grid, Renew. Sustain. Energy Rev. 91 (2018) 90–108.
[23] K. Alanne, S. Sierla, An overview of machine learning applications for smart buildings, Sustainable Cities Soc. 76 (2022) 103445, http://dx.doi.org/10.1016/j.scs.2021.103445.
[24] F. Cicirelli, A. Guerrieri, G. Spezzano, A. Vinci, IoT Edge Solutions for Cognitive Buildings, in: Internet of Things, Springer Cham, 2023, http://dx.doi.org/10.1007/978-3-031-15160-6.
[25] V. Ferraro, J. Settino, Evacuation and smart exit sign system, in: F. Cicirelli, A. Guerrieri, C. Mastroianni, G. Spezzano, A. Vinci (Eds.), The Internet of Things for Smart Urban Ecosystems, Springer, 2019, pp. 363–383, http://dx.doi.org/10.1007/978-3-319-96550-5_15.
[26] R. Zinno, S. Artese, G. Clausi, F. Magarò, S. Meduri, A. Miceli, A. Venneri, Structural health monitoring (SHM), in: F. Cicirelli, A. Guerrieri, C. Mastroianni, G. Spezzano, A. Vinci (Eds.), The Internet of Things for Smart Urban Ecosystems, Springer, 2019, pp. 225–249, http://dx.doi.org/10.1007/978-3-319-96550-5_10.
[27] X. Ding, W. Du, A. Cerpa, Octopus: Deep reinforcement learning for holistic smart building control, in: Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation, 2019, pp. 326–335.
[28] L. Scarcello, F. Cicirelli, A. Guerrieri, C. Mastroianni, G. Spezzano, A. Vinci, Pursuing energy saving and thermal comfort with a human-driven DRL approach, IEEE Trans. Hum.-Mach. Syst. (2022) 1–13, http://dx.doi.org/10.1109/THMS.2022.3216365.
[29] A. Jindal, B.S. Bhambhu, M. Singh, N. Kumar, K. Naik, A heuristic-based appliance scheduling scheme for smart homes, IEEE Trans. Ind. Inform. 16 (5) (2019) 3242–3255.
[30] F. Cicirelli, V. D'Agostino, A.F. Gentile, E. Greco, A. Guerrieri, L. Rizzo, G. Scopelliti, Intelligent load scheduling in cognitive buildings: A use case, in: IoT Edge Solutions for Cognitive Buildings, Springer, 2022, pp. 305–328.
[31] W. Liao, L. Xie, J. Xi, Y. Bai, T. Zhang, Y. Wu, Intelligent parking lot control system based on alibaba cloud platform and machine learning, in: 2021 6th International Conference on Intelligent Computing and Signal Processing, ICSP, IEEE, 2021, pp. 908–911.
[32] Y. Himeur, K. Ghanem, A. Alsalemi, F. Bensaali, A. Amira, Artificial intelligence based anomaly detection of energy consumption in buildings: A review, current trends and new perspectives, Appl. Energy 287 (2021) 116601.
[33] N. Cauchi, K. Macek, A. Abate, Model-based predictive maintenance in building automation systems with user discomfort, Energy 138 (2017) 306–315, http://dx.doi.org/10.1016/j.energy.2017.07.104.
[34] L.C. Tagliabue, I. Yitmen, Special issue cognitive buildings, Appl. Sci. 12 (5) (2022) 2460.

[35] L. Yu, S. Qin, M. Zhang, C. Shen, T. Jiang, X. Guan, A review of deep reinforcement learning for smart building energy management, IEEE Internet Things J. 8 (15) (2021) 12046–12063.
[36] G. Gao, J. Li, Y. Wen, DeepComfort: Energy-efficient thermal comfort control in buildings via reinforcement learning, IEEE Internet Things J. 7 (9) (2020) 8472–8484.
[37] Y.R. Yoon, H.J. Moon, Performance based thermal comfort control (PTCC) using deep reinforcement learning for space cooling, Energy Build. 203 (2019) 109420.
[38] D. Domínguez-Barbero, J. García-González, M.A. Sanz-Bobi, E.F. Sánchez-Úbeda, Optimising a microgrid system by deep reinforcement learning techniques, Energies 13 (11) (2020) 2830.
[39] J. Lu, P. Mannion, K. Mason, A multi-objective multi-agent deep reinforcement learning approach to residential appliance scheduling, IET Smart Grid (2022).
[40] F. Almaguer-Angeles, J. Murphy, L. Murphy, A.O. Portillo-Dominguez, Choosing machine learning algorithms for anomaly detection in smart building iot scenarios, in: 2019 IEEE 5th World Forum on Internet of Things, WF-IoT, IEEE, 2019, pp. 491–495.
[41] S. Nagarathinam, V. Menon, A. Vasan, A. Sivasubramaniam, Marco-multi-agent reinforcement learning based control of building hvac systems, in: Proceedings of the Eleventh ACM International Conference on Future Energy Systems, 2020, pp. 57–67.
[42] J. Li, W. Zhang, G. Gao, Y. Wen, G. Jin, G. Christopoulos, Toward intelligent multizone thermal control with multiagent deep reinforcement learning, IEEE Internet Things J. 8 (14) (2021) 11150–11162.
[43] L.C. Tagliabue, F.R. Cecconi, S. Rinaldi, A.L.C. Ciribini, Data driven indoor air quality prediction in educational facilities based on IoT network, Energy Build. 236 (2021) 110782.
[44] S. Rinaldi, A. Flammini, L.C. Tagliabue, A.L.C. Ciribini, An IoT framework for the assessment of indoor conditions and estimation of occupancy rates: Results from a real case study, Acta Imeko 8 (2) (2019) 70–79.
[45] M. Amadeo, F. Cicirelli, A. Guerrieri, G. Ruggeri, G. Spezzano, A. Vinci, COGITO: A platform for developing cognitive environments, in: IoT Edge Solutions for Cognitive Buildings, Springer, 2023, pp. 1–22.
[46] M. AbdelBaky, M. Zou, A.R. Zamani, E. Renart, J. Diaz-Montes, M. Parashar, Computing in the continuum: combining pervasive devices and services to support data-driven applications, in: IEEE 37th International Conference on Distributed Computing Systems, ICDCS, 2017, pp. 1815–1824.
[47] N.R. Jennings, On agent-based software engineering, Artificial Intelligence 117 (2) (2000) 277–296, http://dx.doi.org/10.1016/S0004-3702(99)00107-1.
[48] M. Wooldridge, An Introduction to Multiagent Systems, John wiley & sons, 2009.
[49] F. Cicirelli, A. Guerrieri, G. Spezzano, A. Vinci, An edge-based platform for dynamic smart city applications, Future Gener. Comput. Syst. 76 (2017) 106–118.
[50] R. Jia, M. Jin, K. Sun, T. Hong, C. Spanos, Advanced building control via deep reinforcement learning, Energy Procedia 158 (2019) 6158–6163.
[51] D.C. Mocanu, E. Mocanu, P. Stone, P.H. Nguyen, M. Gibescu, A. Liotta, Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science, Nature Commun. 9 (1) (2018) 2383.
[52] H.H. Bosman, G. Iacca, A. Tejada, H.J. Wörtche, A. Liotta, Spatial anomaly detection in sensor networks using neighborhood information, Inf. Fusion 33 (2017) 41–56, http://dx.doi.org/10.1016/j.inffus.2016.04.007.
[53] A. Bochkovskiy, C.-Y. Wang, H.-Y.M. Liao, Yolov4: Optimal speed and accuracy of object detection, 2020, arXiv preprint arXiv:2004.10934.
[54] E. Greco, G. Spezzano, Human-centered reinforcement learning for lighting and blind control in cognitive buildings, in: IoT Edge Solutions for Cognitive Buildings, Springer, 2023, pp. 285–303.
[55] A. Franco, E. Schito, Definition of optimal ventilation rates for balancing comfort and energy use in indoor spaces using CO2 concentration data, Buildings 10 (8) (2020) URL https://www.mdpi.com/2075-5309/10/8/135.
[56] ISO/IEC. 11801-2:2017 information technology–Generic cabling for customer premises, 2017, Available online: https://www.iso.org/standard/66183.html. (Last Accessed 21 January 2023).
[57] Y. Su, D. Feng, Y. Hua, Z. Shi, Predicting response latency percentiles for cloud object storage systems, in: IEEE 46th International Conference on Parallel Processing, ICPP, 2017, pp. 241–250.
[58] E. Stefanov, E. Shi, Multi-cloud oblivious storage, in: Proceedings of the ACM SIGSAC Conference on Computer & Communications Security, 2013, pp. 247–258.
[59] Q. Cao, A.E. Irimiea, M. Abdelfattah, A. Balasubramanian, N.D. Lane, Are mobile DNN accelerators accelerating dnns? in: Proceedings of the 5th International Workshop on Embedded and Mobile Deep Learning, 2021, pp. 7–12.
[60] W. Saad, M. Bennis, M. Chen, A vision of 6G wireless systems: Applications, trends, technologies, and open research problems, IEEE Netw. 34 (3) (2019) 134–142.
[61] F. Cicirelli, A. Guerrieri, A. Mercuri, G. Spezzano, A. Vinci, ITEMa: A methodological approach for cognitive edge computing IoT ecosystems, Future Gener. Comput. Syst. 92 (2019) 189–197.
[62] F. Cicirelli, G. Fortino, A. Guerrieri, G. Spezzano, A. Vinci, Metamodeling of smart environments: from design to implementation, Adv. Eng. Inform. 33 (2017) 274–284.