Distributed artificial intelligence for edge computing

Politecnico
di Bari

Department of Electrical and Information Engineering
Electrical and Information Engineering
Ph.D. Program
SSD: ING-INF/05 – Information Processing Systems
Final Dissertation

Distributed Artificial Intelligence for Edge Computing

---

by
Corrado Fasciano

Supervisors:
Prof. Michele Ruta
Prof. Floriano Scioscia
Eng. Felice Vitulano

Coordinator of Ph.D. Program:
Prof. Mario Carpentieri

Course n. 35, 01/11/2019 – 31/01/2023

Politecnico
di Bari

Department of Electrical and Information Engineering
Electrical and Information Engineering
Ph.D. Program
SSD: ING-INF/05 – Information Processing Systems
Final Dissertation

Distributed Artificial Intelligence for Edge Computing

---

by
Corrado Fasciano

Referees:
Prof. Devis Bianchini
Prof. Raffaele Gravina

Supervisors:
Prof. Michele Ruta
Prof. Floriano Scioscia
Eng. Felice Vitulano

Coordinator of Ph.D. Program:
Prof. Mario Carpentieri

Course n. 35, 01/11/2019 – 31/01/2023

Contents

1 Introduction **1**

2 Background **5**

2.1 Edge Computing 5

2.1.1 Edge Intelligence 7

2.1.2 Current issues and limitations 11

2.2 Knowledge Representation and Reasoning 17

2.2.1 Description Logics 17

2.2.2 Reasoning services 20

2.2.3 State of the art 23

2.3 Argumentation basics 26

2.3.1 Argumentation semantics 28

2.3.2 Generalizations of Argumentation Framework 35

3 Cloud-Edge Artificial Intelligence **43**

3.1 State of the art 44

3.2 Osmotic Cloud-Edge architecture 46

3.2.1 Microservices 50

3.2.2 Technologies 52

3.3 Case study 55

3.3.1 Prototype implementation 56

3.4 Experimental evaluation 63

4 Deductive Argumentation Framework **72**

4.1 State of the art 74

4.2 Proposed framework 78

4.2.1 Interpretable Bipolar Weighted Argumentation 78
4.2.2 Propagation-based ranking semantics 81
4.3 Case study 88
4.4 Experiments 98

5 Conclusions and perspectives **113**

List of Figures

2.1 Cloud Computing paradigm 6
2.2 Edge computing paradigm 7
2.3 Six levels of EI [146] 9
2.4 $\mathcal{F}_1$: AF example 27
2.5 $\mathcal{F}_2$: AF example for Ranking-based semantics 32
2.6 BAF example 37
2.7 $\mathcal{W}_1$: WAF example 40
2.8 BWAF legend: b_1 attacks a_1, b_2 supports a_2 41

3.1 Osmotic Cloud-Edge architecture 47
3.2 Components of the proposed prototype 56
3.3 Message Broker/Orchestrator and Edge devices in the platform prototype 57
3.4 Sequence diagram for edge-side prediction 62
3.5 Time required for data import 64
3.6 Dataset transfer times 70

4.1 Relation type definition algorithm between pairs of arguments 79
4.2 Deductive argumentative agent architecture 89
4.3 SC2 case study scenario 90
4.4 Ontology excerpt for the StarCraft II agent 92
4.5 $\mathcal{G}_1$: BWAF of the first deductive argumentative stage 93
4.6 $\mathcal{G}_2$: BWAF of the second deductive argumentative stage 97
4.7 Argument quartile change in acceptability rankings due to graph pruning 109

4.8 Argument position shifts in acceptability rankings due to graph pruning 111

List of Tables

2.1 Syntax and semantics of $\mathcal{ALN}$ 19
2.2 Correspondence between OWL RDF/XML and DL syntax . . 20
2.3 Match classes 24
2.4 Count of discussions for the arguments in $\mathcal{F}_2$ 34
2.5 Burden numbers for the arguments in $\mathcal{F}_2$ 35

3.1 Comparative table of frameworks (✓: supported, ✗: not supported) 46
3.2 Reference COTS tools 52
3.3 Attributes of control message 60
3.4 Attributes of data messages 61
3.5 Training: Network activity 65
3.6 Training time and validation results 66
3.7 Training on Edge node: reduced datasets 67
3.8 Prediction: Network activity 67
3.9 Prediction: time and latency 67
3.10 Microservice deployment performance 70

4.1 Acceptability Ranking on $\mathcal{G}_1$ 96
4.2 Acceptability Ranking on $\mathcal{G}_2$ 98
4.3 Performance results on $\mathcal{G}_1$ and $\mathcal{G}_2$ 100
4.4 Argumentative graph performance results 101
4.5 Bipolar weighted graph configurations 102
4.6 Performance on complete graphs 102
4.7 Performance metrics for total ranking 104
4.8 Performance metrics for the first 5 nodes of the ranking 105

4.9 Performance metrics for the last 5 nodes of the ranking 105
4.10 First and last node comparison 106
4.11 Quartile shifts in argument rankings due to pruning 108
4.12 Argument position shifts due to pruning 110
4.13 Impact of graph pruning on processing time and memory usage peak 110

Abstract

In an increasingly wide range of work and personal life settings, sensors and micro-devices embedded into objects generate continuous data streams with high volume, velocity and heterogeneity. Such data can be analyzed to detect and infer knowledge about phenomenons and events of interest. By learning from data, Artificial Intelligence (AI) methods enable automating an ever larger amount of activities and decision-making tasks with comparable or better proficiency than human experts. Big Data applications based on pervasive Internet of Things (IoT) deployments are now a well-established reality: they feed large Machine Learning (ML) models, which are trained exploiting the huge computational resources of cloud computing infrastructures to offer increasingly accurate prediction capabilities on fresh data. However, the increasing miniaturization of IoT devices equipped with highly accurate sensors enables novel Cyber-Physical Systems (CPSs) with tight feedback loops coupling computation, communication and control tasks. CPS applications are expanding in sensitive fields like high-precision manufacturing, telemedicine, and self-driving vehicles. Those scenarios require real-time response, high computational and bandwidth efficiency, cost-effectiveness to support business scalability and strict data privacy constraints. For this reason, classical cloud-based approaches are progressively integrated with the Edge Computing (EC) architectural model, which distributes significant processing and storage resources at the edge of the local network, in closer proximity to field devices and sensors. This paradigm allows AI-based IoT applications to scale even more, as models are trained with massive amounts of data generated by large deployments of micro- and nano-devices, and ML inference achieves ever greater accuracy. In this context, the Edge Intelligence paradigm –which promotes the integration of EC and AI– is increasingly adopted to execute inference on data at the border of local networks, employing models trained in the cloud. The next logical step in Cloud-Edge AI cooperation is to enable training tasks on edge nodes as well. However, as of now flexible approaches to combine Edge Intelligence with cloud infrastructures, allowing dynamic

migration of training and inference tasks, are not available yet. This thesis proposes a Cloud-Edge AI microservice architecture, based on *Osmotic Computing* principles. A full pipeline consisting in data collection from local devices, preprocessing, AI model training and inferencing is supported either on the edge, on the cloud or both, exploiting computational resources opportunistically based on device status, available bandwidth, and application requirements on latency, prediction accuracy, and privacy. To demonstrate the feasibility of the proposed framework, a prototype has been realized with commodity hardware leveraging open-source software technologies. It has been used in a small-scale intelligent manufacturing case study, carrying out experiments on elapsed time and network activity in (i) data gathering, (ii) AI model training and validation, and (iii) prediction tasks. Obtained results validate the key benefits of the approach.

In addition to architectural aspects, open issues still limiting the full applicability of EC include the heterogeneity of devices, services, and information that arise in pervasive contexts, the integrity of the gathered data, and the trustworthiness and dependability of autonomous decisions. The *Semantic Web of Things* (SWoT), coalescing the Semantic Web and IoT paradigms, has been proposed to overcome these problems. In SWoT environments, the dynamic exchange of knowledge fragments expressed in logic-based formalisms in volatile wireless networks of independent agents enables decentralized collaborative service discovery, autonomous decision and user decision support. A relevant problem in such scenarios consists in evaluating agreements and disagreements about knowledge produced by different interacting agents, in order to possibly reconcile conflicts and determine the best overall outcome to accomplish distributed coordination. In AI, *Argumentation* is recognized as a powerful formalism to negotiate and solve disagreements within a group of agents, which convey knowledge represented as a constellation of arguments and counterarguments. The argumentation literature provides a wealth of frameworks for agent decision-making and coordination. Nevertheless, few proposals leverage Semantic Web languages and technologies, which can provide a well-known formal model for arguments, well-studied inference algorithms for the assessment of argument relations and approaches to evaluate argument acceptability. This thesis presents a novel

Bipolar Weighted Argumentation Framework, where arguments are modeled as Description Logics concept expressions in Web Ontology Language (OWL) 2, and their relations are assessed via semantic matchmaking, leveraging non-standard inference services with logic-based outcome explanation. Argument acceptability is computed via a novel propagation-based ranking semantics, which supports argument cycles and information fading. In order to make the proposal suitable for pervasive semantic agents in resource-constrained devices, optimizations in argument assessment and ranking evaluation are adopted, while a graph simplification approach via pruning is proposed and experimentally tested as a tunable trade-off between computational resource usage and accuracy of results. Validation of the approach has been carried out by means of a prototypical implementation of a player agent for the *StarCraft II* real-time strategy game, whose environment allows simulating the complexities of real CPS scenarios.

Chapter 1

Introduction

The *Internet of Things* (IoT) [78] vision is increasingly enabling the diffusion of micro- and nano-devices incorporated into objects dipped in everyday environments and capable of collecting, storing, processing and exchanging non-negligible amounts of information. Cyber-Physical Systems (CPSs) integrate computational, physical sensing/actuation, environmental, and human components to implement, control, and automate complex operations in the real world by leveraging IoT technological advancements [116]. Consequently, huge quantities of data from natural and human-made settings and processes can be gathered and analyzed.

At the same time, Artificial intelligence (AI) methods and technologies are evolving with fast pace and diversification. They allow simulating extremely accurate rational behavior in machines by learning from data and outperforming skilled professional humans in an increasing range of tasks [32]. Current AI trends make use of ever-larger information corpora to train Machine Learning (ML) models and offer inference (*i.e.*, prediction) capabilities on measurements with increasing accuracy. Early IoT-based AI solutions uploaded all data to cloud computing infrastructures for model training and inference. Conversely, the rising proliferation of powerful processing devices on nodes at the boundary of local area networks has led to a significant increase in the use of *Edge Computing* (EC) [2] for those purposes. The fundamental goal of EC is to move computing and communication resources from the cloud to the edge of networks, in order to provide services and carry

out quick computations while preventing conspicuous communication latency and facilitating quicker replies for end users.

Due to the relevance of the EC and AI trends, the new *Edge Intelligence* (EI) paradigm [44, 88] promotes the confluence of the two. Commonly, EI provides the ability to run AI models on Edge devices while still leaving the computationally more expensive task of training AI models to powerful cloud datacenters. The next evolutionary step has been recognized as collaborative *Cloud-Edge Intelligence*, where model training and prediction tasks can be carried out either in edge or cloud nodes, depending on application requirements and resource availability [146]. However, due to the complexity of dynamically managing resources and services across the Edge and Cloud tiers, effective practical architectures and solutions are challenging to realize. Osmotic Computing (OC) [135] is an innovative methodology aimed at highly distributed and federated environments, powered by advanced EC capabilities. In OC, dynamic orchestration enables the automatic deployment and elastic migration of microservices from edge to cloud infrastructure nodes and vice versa, in order to optimize availability and performance with respect to variable workloads, network topology, and mobility of devices and resources.

This thesis presents a novel Cloud-Edge AI microservice architecture for IoT-oriented CPSs based on the OC paradigm. It supports gathering data streams from local cyber-physical devices, preprocessing them and performing AI model training and inference in ML classification and regression problems. Most notably, the same AI microservices –encapsulated in *containers*– can be deployed either to edge nodes or to cloud infrastructure *opportunistically*, *i.e.*, exploiting resources available in the neighborhood of the task to be completed and in the current time frame [79]. Edge AI can grant lower prediction latency and turnaround time, in addition to inherently higher data privacy; conversely, cloud AI can maximize model accuracy and provide further large-scale analytics capabilities. By supporting hybrid Cloud-Edge AI solutions, the proposed platform aims to offer the best of both worlds to end users.

Although EC is suitable for solving the typical problems of centralized Big Data solutions concerning communication latency, security, and violation of privacy in data transmission, some open questions persist, limiting

its full applicability: the heterogeneity of devices, services and information that emerge in pervasive contexts, the truthfulness of the information collected, and the reliability of the decisions autonomously recommended by the devices. While the Semantic Web community [20] has provided standard frameworks, languages and tools to describe resources with semantic metadata, query them and reason upon them, the communication and coordination aspects have usually been left to individual applications or vertical interest groups. The *Semantic Web of Things* (SWoT) paradigm [112] aims at integrating Semantic Web and Internet of Things technologies in pervasive computing scenarios. SWoT environments support automated reasoning on so-called *ubiquitous knowledge bases* (u-KBs), comprising individual knowledge fragments physically disseminated among heterogeneous smart objects interconnected in mobile ad-hoc networks (MANETs). Knowledge exchange occurs via machine-to-machine interactions based on collaborative protocols. These dynamic interactions can be considered as an ongoing *dialogue* among independent agents [83]. This means agreements or conflicts about object assertions can happen. Particularly, disagreements should be solved in order to achieve decentralized coordination and decision-making in wireless networks. The branch of AI called *argumentation* provides a principled approach to deal with these situations. Argumentation studies the general problem of multi-agent discussion. Arguments represent knowledge asserted by agents; they are defeasible, *i.e.*, their validity can be disputed by other arguments, if new information becomes available. A set of arguments is organized as a constellation, *i.e.*, a graph, where edges represent relations between pairs of arguments. *Abstract Argumentation* (AA) [49] evaluates the acceptability of each argument only according to those relations, abstracted from the content of the arguments themselves. *Structured Argumentation* (SA) [21], instead, adopts formal models to represent arguments and studies automated methods to assess relations. The SWoT knowledge interchange fits very well the classical argumentation paradigms, nevertheless mature proposals exploiting rich *Knowledge Representation and Reasoning* (KRR) frameworks such as the ones investigated within the Semantic Web initiative are lacking in the SA literature. Their adoption can expand the possibilities for effective argumentation frameworks among pervasive agents, while at the same time

posing challenges concerning the characterization of the formal properties of such new frameworks as well as their computational feasibility and practical applicability.

In this perspective, the thesis proposes a novel general-purpose SA framework, exploiting Description Logics-based KRR for argument representation, relation appraisal and acceptability evaluation. Information generated and shared by an agent are expressed as *Web Ontology Language* 2 annotations [96]. The approach is compatible with Dung-style AA [49], where each argument is represented by a semantic annotation. Specifically, it refers to a *Bipolar Weighted Argumentation Framework* (BWAF) [97], which allows a fine-grained characterization of relations by means of weights representing their type (attack or support) and strength. Relation appraisal leverages non-standard, non-monotonic inference services *Concept Contraction*, *Concept Abduction* and *Concept Bonus* [104]. A novel *propagation-based gradual semantics* –which also handles *cyclic* argumentative graphs– computes an acceptability score for each argument and identifies the most and least acceptable ones, enabling autonomous decision and user decision support in Multi-Agent Systems (MAS) [103]. In fact, the framework has been integrated with the Real-Time Strategy (RTS) game engine *StarCraft II* (SC2) in order to select the most appropriate strategic plan in a given instant of the battle.

The remainder of this dissertation is organized as in what follows.

Chapter 2 introduces the research context and recalls the technological background for the work.

Chapter 3 provides in detail a functional description of the Cloud-Edge AI microservice architecture. An intelligent manufacturing case study is presented to allow a better understanding of the proposal and the realized prototype, along with performance evaluation experiments.

Chapter 4 describes the proposed deductive argumentative framework. Validation of the approach has been carried out by means of a prototypical implementation of a player agent for SC2. Details about experimental results are provided as well.

Chapter 5 concludes the dissertation by summarizing the main contributions and outlining open research perspectives.

Chapter 2

Background

This chapter presents an overview of the state of the art trends in the Edge Computing. Compared to conventional cloud solutions, by placing an increasing amount of processing and storage resources close to data sources, in this paradigm processing times and bandwidth usage are typically reduced. Limitations and present concerns that restrict the EC full usability are analyzed, in order to outline the problems tackled in the research work for this thesis. Moreover, basics are discussed on the languages and tools for Knowledge Representation and Reasoning in the Semantic Web of Things and on Argumentation theory. Their visions and foundational technologies are described, in order to provide a theoretical and technical background for the subsequent chapters.

2.1 Edge Computing

In modern IoT applications, information generation mainly takes place at the edge of the network and response times are required to be increasingly short. The data processing phase is also gradually moving to the edge, with a consequent positive impact on issues concerning performance efficiency and security.

So far, scheduling all processing activities on the cloud has proved to be an efficient methodology for analyzing large amounts of data, as the computing resources available on cloud infrastructures far exceeded the computational

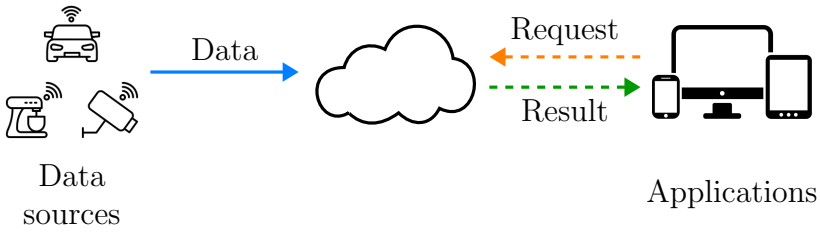


Figure 2.1: Cloud Computing paradigm

capabilities of common smart objects adopted at the edge of the network. However, the continuous and rapid growth of data production rate and the progressive consumption of currently available network resources make data transmission rate the real bottleneck of cloud solutions [122]. Figure 2.1 shows the architectural scheme of a conventional cloud computing solution. In this architecture, end devices act as *data sources*, generating raw data from processes in the field and transferring them to the cloud platform, while entities known as *applications* issue requests to the cloud in order to retrieve and use information on behalf of users.

This type of architecture does not appear the most suitable solution for IoT applications where (i) latency is mission-critical, (ii) the large amount of data produced causes an excessive use of bandwidth at the edge of the network and (iii) there are stringent data privacy protection constraints. Some architectural paradigms, such as micro-datacenters [1], cloudlet [82] and fog computing [12], have been introduced to compensate these shortcomings of cloud computing in processing the data produced by IoT device networks. Moreover, in highly dynamic scenarios characterized by unpredictable device and resource availability, *opportunistic computing* [39] has emerged as a paradigm oriented to build platforms which are capable of maximizing the exploitation of available resources in the environment to execute distributed computing tasks. The main challenge with opportunistic computing is making effective use of volatile connections to ensure that data is easily accessible and that collaborative computing services are provided to both applications and users. In this perspective, middleware services must hide disconnections and delays as well as handle a variety of computing resources, services, and data.

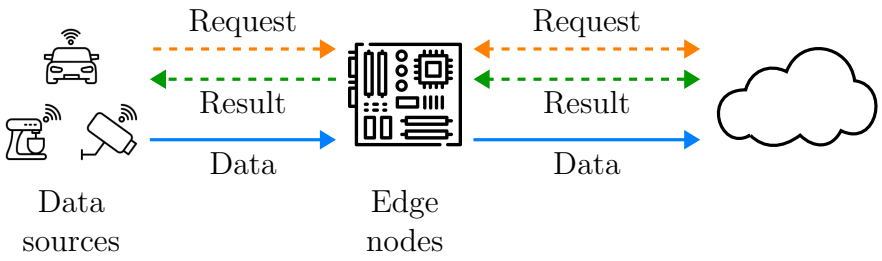


Figure 2.2: Edge computing paradigm

Edge Computing (EC) is a new paradigm in which substantial computing and storage resources are placed at the local network's edge, in close proximity to field devices and sensors [110]. The term "edge" denotes any computing and network resource along the path between the data source and the cloud data center: for example, in the healthcare sector a smartphone is the edge between wearable devices and a cloud service, while a gateway in a home automation network is the edge between smart sensors and a cloud platform. Basically, the goal of the EC paradigm is to perform computational tasks closer to data sources.

A generic EC architecture is depicted in Figure 2.2. In this model, an intermediate layer of edge nodes is deployed between end devices and the cloud. An edge node can (i) collect data from sensors and field devices, (ii) perform pre-processing activities and/or propagate them to the cloud, and (iii) request services and exchange contents with the cloud. In an IoT application design, leveraging edge nodes ensures the offloading, storing, caching, and processing of data, as well as distributing the computational load within the local network.

2.1.1 Edge Intelligence

Edge Computing is considered a disruptive and rapidly growing architectural paradigm in today's technological era [71]. Placing computing and communication resources at the edge of the network where device generate data reduces communication latency and increases responsiveness for end users in pervasive scenarios ranging from digital healthcare to smart manufacturing [3, 99]. Meanwhile, Artificial Intelligence (AI), in its most various tech-

niques, is strongly modifying our daily lives, making machines increasingly capable of planning, learning, reasoning, problem solving, knowledge representation, perception, navigation, object manipulation, language processing, social intelligence and content generation, all domains traditionally belonging to human intelligence. Since AI is useful in cyber-physical systems for analyzing huge volumes of data and extracting higher-level information from them, a strong demand to integrate Edge Computing and AI has emerged in recent years, giving rise to the Edge Intelligence paradigm [44, 88]. Edge Intelligence is not the simple combination of Edge Computing and AI, but it covers many concepts and technologies, often interconnected in a complex way.

So far, researchers have conveyed their interests on EI according to different perspectives. The work in [44] proposes a broad vision suggesting the division of EI into two areas: "*AI for edge*" and "*AI on edge*".

- AI for edge focuses on providing a better solution to constrained optimization problems in Edge Computing by adopting powerful AI technologies. Here, the edge is endowed with more intelligence and efficiency by means of AI. Consequently, it can be thought of as Intelligence-enabled Edge Computing (IEC).

- AI on Edge investigates how to execute AI models on edge. This approach enables training of and inference with AI models through a device-edge-cloud synergy and aims at extracting insights from huge and distributed data to satisfy requirements of algorithm performance, cost, privacy, reliability, efficiency, etc.

Currently, there is still no formal definition of Edge Intelligence, albeit in [145] it is defined as the paradigm that allows the local execution of AI algorithms on a terminal device, with data and signals generated by it. Although this is the current most common approach to EI in real-world applications, this definition greatly narrows its scope of applicability. Running computationally expensive algorithms on embedded devices requires considerable computing power. This requirement not only increases the cost of an EI solution, but is also hardly compatible with current end devices, which have limited processing capabilities.

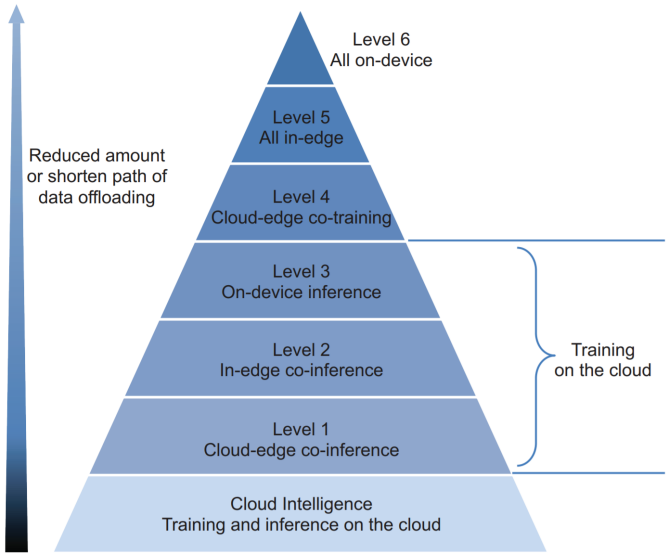


Figure 2.3: Six levels of EI [146]

In light of this, in [146] it is argued that the EI applicability should not be limited to executing AI models exclusively on devices or edge servers. AI models should be run with an edge-cloud coordination allowing to reduce both end-to-end latency and power consumption w.r.t. the local execution approach. These practical benefits promote the integration of a collaborative hierarchy into the design of effective EI solutions. Most frequently, only the inference phase of an AI model takes place at the edge of the network, since the training phase is delegated to cloud datacenters where much larger computational resources are available. However, this implies massive amounts of training data have to be moved to the cloud, leading to excessive communication costs and data privacy concerns. Data preprocessing and reduction at the edge are useful methods to handle these large amounts of data efficiently before storage, transmission and analysis. Nevertheless, scalability and latency are still open research problems [98].

Conversely, Edge Intelligence takes full advantage of the data and resources available across the hierarchy of terminal devices, edge nodes and cloud datacenters to optimize the overall training performance and subsequently exploit trained models for inference. Therefore, using the EI paradigm does not necessarily mean that the AI model is fully trained or

inferred at the edge of the network, but that these operations can take place in a coordinated cloud-edge fashion through data offloading. Particularly, according to the amount and path length of data offloading, Zhou *et al.* [146] classify Edge Intelligence in six levels, as depicted in Figure 2.3 and explained in what follows.

- **Cloud Intelligence:** training the AI model and inferencing fully in the cloud.
- **Level 1 – Cloud-Edge co-inference and Cloud training:** training the AI model in the cloud, but inferencing through edge-cloud cooperation, which entails a partial data offload to the cloud.
- **Level 2 – In-Edge co-inference and Cloud training:** training the AI model in the Cloud, but inferencing at the network edge, by completely or partially offloading the data to the edge nodes or nearby devices.
- **Level 3 – On-Device inference and Cloud training:** training the AI model in the Cloud but inferencing in a fully local on-device manner. This implies that no data would need to be offloaded.
- **Level 4 – Cloud-Edge co-training and inference:** both training and inferencing occur in the edge-cloud cooperative way.
- **Level 5 – All In-Edge:** training and inferencing fully occur at the edge.
- **Level 6 – All On-Device:** both training and inferencing occur on local (field) devices.

As the level of Edge Intelligence increases, the amount and length of the data offloading path decreases. As a result, data privacy increases and transmission latency and bandwidth cost decrease. However, this is achieved at the cost of higher computational latency. Power consumption in the local area network likewise undergoes a variable trade-off between communication and processing. This analysis shows there is no "best" level in absolute, but the "best level" should be decided for each application by jointly taking into

account a variety of factors, including latency, energy efficiency, privacy, and bandwidth costs.

2.1.2 Current issues and limitations

The adoption of the Edge Computing architectural paradigm supports the localized processing and storage of data at the peripheral nodes of the network, in close proximity to the end user. Compared to classic cloud solutions with centralized servers, EC promotes an evolution towards more distributed approaches and consequently reduces processing latencies by partitioning the computational load, limits energy consumption and guarantees flexibility in development. In view of the intrinsic potential and the application advantages of this architectural model, there are yet some limitations and open research challenges described hereafter.

Naming

A typical EC infrastructure includes a large number of nodes to be managed. From an architectural perspective, on top of the edge nodes several applications are running, each with its own distributed computing architecture and providing a particular service to the end user. As with all computer systems, the *naming scheme* in EC is crucial for programming, addressing, object identification, and data transmission. Unfortunately, a reliable and consistent naming system for this kind of architectural paradigm has not yet been built and standardized. A useful scheme must be able to manage the mobility of objects in a highly dynamic network topology, while ensuring the protection of privacy and security.

Traditional identification mechanisms, such as the Domain Name Service (DNS) and the Uniform Resource Identifier (URI), meet the requirements of most current networks very effectively; however, they are not flexible enough to handle the dynamics of EC architectures. New naming techniques applied to EC include Named Data Networking (NDN) [144] and MobilityFirst [101]. Specifically, the mechanism underlying NDN (i) provides a hierarchical and well-structured addressing scheme for dynamic networks based on content/data, (ii) is straightforward in the management/configuration of

services and (iii) offers good scalability for Edge solutions. Nevertheless, the NDN solutions proposed so far focus on networks based on the Internet Protocol (IP), without offering mechanisms to configure additional proxies for the adaptation and integration of other IoT-oriented communication protocols as Bluetooth [24] or ZigBee [54]. Security is another issue with NDN, since it is very challenging to differentiate hardware information from service providers [37].

MobilityFirst is able to keep the name associated with an edge device separate from its network address by means of a dynamic mapping procedure. This allows supporting the main edge services even when the nodes have highly dynamic mobility. Nevertheless, the main issue of MobilityFirst lies in the difficulty of managing and deploying the services, due to the complicated and unintuitive configuration process.

Privacy and Security

In edge solutions, privacy management and data security are crucial requirements to be fulfilled. A home or work environment equipped with several IoT objects could be the source of large amounts of sensitive information, which could be inferred through data mining and analysis. In such scenarios, security- and privacy-oriented architectures and the support for monitoring and management services represent very important challenges. Moving computing from cloud infrastructures to the edge of the network can contribute effectively in protecting data privacy and security. Pre-processing data –particularly of sensitive nature– at the edge of the network to consolidate information can help reduce the data streams transmitted to the cloud through the Internet, thus mitigating the risks of snooping and tampering [141]. However, several issues are still open:

- The awareness of the community that privacy and security issues play a key role in architectures of this type is still limited [5]. Service providers, system integrators, application developers and end users must be aware that privacy can be violated without warning especially at the edge of the network: for example IP cameras, health monitoring devices and smart home automation equipment can be easily hacked if

not protected properly.

- The ownership of the data collected at the edge can be compromised. Similarly to mobile applications, data collected from disparate smart objects in pervasive contexts is stored, transferred and analyzed by a service provider. However, from a security perspective, it could be better to store information in or close to the location where it is gathered. The data gathered at the network's edge should be maintained on the edge node and possibly transferred to a service provider for processing only after a thorough evaluation by the user or the data protection officer [80]. In fact, during this authorization process, some data deemed particularly sensitive could also be removed from the information transferred over the network.

- The absence of effective tools for managing privacy and security at the data collection points.The majority of end devices are heavily resource-constrained; consequently, in many cases they cannot sustain typical computation-intensive security procedures. An edge node is an optimized venue to deploy IoT security solutions since: (i) it has more resources than IoT end devices; (ii) it can satisfy real-time requirements needed in security design; (iii) the large availability of data collecting from many different sources enhances the accuracy of security algorithms [117]. Furthermore, the mobility of nodes can make the network itself more difficult to protect.

Optimization metrics

The following performance indicators must always be taken into account and optimized while designing an infrastructure based on the EC paradigm.

Latency. Latency is one of the most crucial metrics for assessing the performance of a computing architecture, particularly for real-time applications in pervasive computing. Cloud computing infrastructures have a large amount of computing resources and can handle large and complex workloads, such as image processing or speech recognition, in a relatively short time. However, latency is not determined by computation time alone. Any

delays introduced in end-to-end communications within a Wide Area Network (WAN) can significantly influence the behavior of real-time interactive applications. In order to decrease response times, computation should take place as close as possible to the devices which collect data and requests. For example, in a manufacturing plant, an edge node can be leveraged to locally process the photos of the manufactured product, for automated quality inspection. Then, only faulty components' information can be sent to the cloud, instead of all products'. However, the physical layer closest to the data may not always be an optimal solution, due to insufficient computational capabilities, especially when running concurrent tasks. An optimal trade-off would be achieved by appropriately balancing the available amount of computing resources and the maximum acceptable waiting time for an application [6].

Bandwidth. In addition to latency, bandwidth impacts transmission times, especially for applications that transfer large amounts of data. If the workload can be handled at the network edge, latency is greatly improved compared to running in the cloud. At the same time, the required bandwidth is reduced and the transmission reliability is improved. Furthermore, in case an edge node does not have the computational capabilities to fulfill a specific computing request, it is always possible to carry out some form of data preprocessing (*e.g.*, data integration, data normalization, data transformation, data reduction, feature selection, instance selection) [58, 98] to decrease the amount of information to be transferred to the cloud.

Energy. Energy represents the most valuable resource for battery-powered wireless nodes in IoT-based pervasive computing. For an endpoint, offloading workload to the edge can be deemed as an energy-saving method [89]. However, in order to assess whether it is more efficient to (partially) transfer the workload to the edge or process it locally, the trade-off between the amount of energy used for computation and for information transmission must be taken into account. In general, the amount and type of required computing resources, the intensity of the network signal, the size of the data to be processed and the available bandwidth should be considered first to produce an objective estimate of the energy impact required by a specific task [67]. Edge Computing should only be employed if the overhead of transmitting

data is less than the cost of local processing. However, considering the entire network architecture rather than just the endpoints involved in the transmission, the total energy consumption is represented by the sum of the energy cost of each device used from the data collection stage to processing phase; the per-device energy cost can be estimated as local computation cost plus transmission cost [143]. In this case, the optimal load allocation strategy may change, depending on whether a local node is busy with other activities or not. In the first case, the computational task is progressively transferred to the edge (or cloud) resources and the multi-hop data transmission could increase the overload of network nodes, causing higher power consumption.

Cost. Reduced cost is among the benefits that encourage companies to invest in EC technologies and architectures. In 2021 *Gartner* predicted that in 2022 over 50% of enterprise-generated data will have been created and processed outside a traditional data center or cloud [43]. Currently, even though the expensive cloud datacenters have been replaced by using on-demand pay-per-use IT resources, the operating costs for companies adopting this architectural model are still high. In an EC solution, the costs essentially concern the infrastructure *i.e.*, sensors, compute systems, and network [132]. In fact, using the local area network for data processing provides enterprises higher bandwidth and storage at lower costs compared to cloud computing. Additionally, since processing takes place at the edge, less data needs to be transferred to the cloud or data center for further processing. As a result, the amount of data that must travel as well as the entire solution's price are reduced.

AI integration

The continuous proliferation of innovative AI/ML methodologies has prompted the scientific community to merge AI with Edge Computing into the Edge Intelligence paradigm [88]. EI offers several benefits:

- pre-processing raw data at the network edge decreases the data dimensionality to be transferred to the Cloud for deeper analysis and reduces bandwidth requirements;
- heterogeneous sensor data fusion algorithms at the network edge lowers

the workload in cloud platforms;

- using the computing capabilities of edge nodes helps to accelerate the response of distributed AI algorithms in Big Data scenarios.

Nevertheless, the EI paradigm increases complexity of edge computing in non-trivial ways: [139]

- the introduction of structured solutions and paradigms to integrate ML in a cloud-edge continuum;
- the need to adapt AI/ML algorithms for distributed computing and distributed storage systems;
- the administration of computing and memory resources required for model training and prediction on data, as well as for training and prediction services themselves.

This thesis proposes an innovative general-purpose Cloud-Edge Intelligence distributed framework (see Chapter 3) to handle the aforementioned issues. It employees an Osmotic Computing paradigm-compliant microservice architecture to enable opportunistic resource exploitation by means of dynamic distribution of service modules to different devices at the edge of the network and/or in the cloud. The ML model training and prediction activities can be carried out in edge or cloud nodes, as well as through cloud-edge collaboration, which is a key feature of the approach.

Further open questions limiting a wider applicability of EC solutions persist: the heterogeneity of devices, services and information that arise in pervasive contexts; the truthfulness of the collected information; the reliability of the decisions autonomously suggested by pervasive devices. This work addresses these issues by defining an innovative framework that integrates Knowledge Representation and Reasoning with Argumentation in a Semantic Web of Things perspective (Chapter 4). In SWoT environments, knowledge exchange among intelligent objects can be considered as an ongoing dialogue among independent agents: in this scenario, conflicts about assertions can happen due to the variability of data and volatility of devices. They could be solved to accomplish decentralized coordination by exploiting

general-purpose formalisms as the argumentation theory. In the following, preliminary notions regarding the Description Logics-based KRR and Argumentation are recalled to make the work self-contained.

2.2 Knowledge Representation and Reasoning

In ubiquitous and pervasive contexts, intelligent software agents running on personal devices can extract, process, and exchange meaningful information fragments in order to enable adaptive context-aware behaviors in many different applications. As agents interact with other agents and with the surrounding environment in a typical perception-decision-action-evaluation loop, they must be able to integrate detected and received information and manage possible agreements and disagreements on perceptions in a collaborative and coordinated way. Furthermore, high degrees of autonomic capability are required in order to trigger actions or make interventions on the environment according to the detected context without human interaction. Knowledge Representation and Reasoning techniques and technologies allow information modelling based on formal and rigorous interpretation of its meaning (semantics). This enables not only greater interoperability across different hardware/software platforms, but also reasoning tasks *i.e.*, inferences, to extract new implicit insight from information explicitly asserted in a Knowledge Base (KB). Among the many available KRR languages and tools, those born from the Semantic Web initiative have gained widespread acceptance with well-known algorithmic properties and highly optimized implementations. In the following subsections, Description Logics family and reasoning tasks relevant for this work are recalled in detail.

2.2.1 Description Logics

Description Logics (DLs) [13] are a family of Knowledge Representation (KR) languages in a decidable fragment of First Order Logic (FOL) [30, 48]. DLs allow to represent knowledge by means of:

- *concepts a.k.a. classes*, representing sets of objects;
- *roles a.k.a. properties*, defining relationships between concepts pairs;
- *individuals*, *i.e.*, named instances of classes.

An *ontology* (*a.k.a.* terminology, terminological box, TBox) [125] is composed by two types of *assertions* involving classes and properties: *inclusion*, which allows to define *is-a* relationships between classes ($A \sqsubseteq D$ where A and D are concept expressions); *equivalence*, which allows to give a name to a particular concept expression ($A \equiv D$). Individuals make up the *assertion box* (ABox). A Knowledge Base (KB) consists of a $\langle$TBox, ABox$\rangle$ pair.

DLs are distinguished by the constructors they provide. This work adopts the *Attributive Language with unqualified Number restrictions* ($\mathcal{ALN}$) DL as reference. It provides adequate expressiveness while keeping polynomial complexity, both for standard and non-standard inferences [46]. Constructs of $\mathcal{ALN}$ are reported in what follows and Table 2.1 summarizes syntax and semantics of constructors and assertions in $\mathcal{ALN}$.

- $\top$, *universal concept*. All the objects in the domain.
- $\bot$, *bottom concept*. The empty set.
- A, *atomic concepts*. All the objects belonging to the set A.
- $\neg A$, *atomic negation*. All the objects not belonging to the set A.
- $C \sqcap D$, *intersection*. The objects belonging to both C and D.
- $\forall R.C$, *universal restriction*. All the objects participating in the relation R whose range are all the objects belonging to set C.
- $\exists R$, *unqualified existential restriction*. There exists at least one object participating in the relation R.
- $(\geq nR)$[1], $(\leq nR)$, $(= nR)$[2], *unqualified number restrictions*. Respectively the minimum, the maximum and the exact number of objects participating in the relation R.

[1]Notice that $\exists R$ is equivalent to $(\geq 1R)$.
[2]Notice that $(= nR)$ is a shortcut for $(\geq nR) \sqcap (\leq nR)$.

Table 2.1: Syntax and semantics of $\mathcal{ALN}$

Name	Syntax	Semantics
Top	$\top$	$\Delta^{\mathcal{I}}$
Bottom	$\bot$	$\emptyset$
Intersection	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
Atomic negation	$\neg A$	$\Delta^{\mathcal{I}} \backslash A^{\mathcal{I}}$
Universal quantification	$\forall R.C$	$\{d_1 \mid \forall d_2 : (d_1, d_2) \in R^{\mathcal{I}} \rightarrow \ d_2 \in C^{\mathcal{I}}\}$
Number restrictions	$\geq nR$	$\{d_1 \mid \sharp\{d_2 \mid (d_1, d_2) \in R^{\mathcal{I}}\} \geq n\}$
	$\leq nR$	$\{d_1 \mid \sharp\{d_2 \mid (d_1, d_2) \in R^{\mathcal{I}}\} \leq n\}$
Inclusion	$A \sqsubseteq D$	$A^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
Equivalence	$A \equiv D$	$A^{\mathcal{I}} = D^{\mathcal{I}}$

Web Ontology Language (OWL)

Web Ontology Language (OWL) [96] is a World Wide Web Consortium (W3C) Recommendation extending RDF Schema for creating ontologies and expressing metadata on resources in the Semantic Web. OWL uses the Internationalized Resource Identifier (IRI) for unique resource naming and is based on the Resource Description Framework (RDF) [63] knowledge model. The first version of OWL branched into three different levels of complexity, from the least to the most expressive:

- *OWL Lite*: allows very basic taxonomy and constraint definition;
- *OWL DL*: enables a fairly wide expressiveness while retaining computational tractability. The name indicates a direct correspondence between OWL and Description Logics;
- *OWL Full*: provides the highest level of flexibility and expressiveness, sacrificing computational decidability.

OWL Lite is a subset of *OWL DL*, which is in turn a subset of *OWL Full*. The subset of *OWL DL* elements allowing to express the $\mathcal{ALN}$ DL is presented in Table 2.2 in *RDF/XML* syntax [111].

In pervasive scenarios featured by volatile nodes interacting in an opportunistic fashion in order to achieve a common goal, *OWL Full* is not suitable because of its intractability, while suitable fragments of *OWL DL* provide

Table 2.2: Correspondence between OWL RDF/XML and DL syntax

OWL RDF/XML syntax	**DL syntax**
<owl:Thing>	$\top$
<owl:Nothing>	$\bot$
<owl:Class rdf:ID="C">	C
<owl:ObjectProperty rdf:ID="R">	R
<rdfs:subClassOf>	$\sqsubseteq$
<owl:equivalentClass>	$\equiv$
<owl:disjointWith>	$\neg$
<owl:intersectionOf>	$\sqcap$
<owl:allValuesFrom>	$\forall$
<owl:someValuesFrom>	$\exists$
<owl:maxCardinality>	$\leq$
<owl:minCardinality>	$\geq$
<owl:cardinality>	$=$

a good trade-off between expressiveness and complexity. In this work the domain of interest was modeled using the latest specification of OWL language, *i.e.*, OWL 2 [96]. It supports a variety of syntaxes to read, store and exchange knowledge conceptualization among applications. OWL 2 includes three sublanguages, named *profiles*:

- *OWL 2 EL*, a fragment that has polynomial time reasoning complexity;
- *OWL 2 QL*, designed to enable easier access and query to data stored in databases;
- *OWL 2 RL*, a rule subset of OWL 2.

Each profile fits specific use cases of practical interest and offers a different trade-off between expressiveness and reasoning efficiency. Unlike early OWL sublanguages, OWL 2 profiles are mutually unrelated.

2.2.2 Reasoning services

Polynomial-complexity *structural* reasoning algorithms for $\mathcal{ALN}$ concept descriptions can be exploited in order to enable distributed information aggregation and discovery in resource-constrained agent networks and MASs [113].

To this aim, when an $\mathcal{ALN}$ KB is loaded, it is preprocessed with *unfolding* and *Conjunctive Normal Form* (CNF) *normalization* [104] procedures. Particularly, given a TBox $\mathcal{T}$ and a concept C, unfolding recursively expands references to axioms in $\mathcal{T}$ within the concept expression itself. In this way, $\mathcal{T}$ is not needed any more when executing subsequent inferences. The CNF translation is then obtained by applying a set of pre-defined substitutions. Any concept expression C in CNF can be expressed as:

$$C \equiv C_{CN} \sqcap C_{\leq} \sqcap C_{\geq} \sqcap C_{\forall}$$

with

- C_{CN}: conjunction of (possibly negated) concept names;
- $C_{\leq}$: conjunction of maximum cardinality restrictions, at most one per role;
- $C_{\geq}$: conjunction of minimum cardinality restrictions, at most one per role;
- $C_{\forall}$: conjunction of universal restrictions, at most one per role; fillers are recursively in CNF.

Normalization preserves semantic equivalence with respect to models induced by the TBox; furthermore, CNF is unique (up to commutativity of conjunction operator) [46]. The normal form of an unsatisfiable concept is simply $\bot$.

Given a DL ontology $\mathcal{T}$ and S, R two satisfiable concepts in $\mathcal{T}$, the *satisfiability* and *subsumption* standard inference services provided by DL-based systems [13] can be formalized as follows:

- *Satisfiability*: verifies if the conjunction of S and R is satisfiable w.r.t. the ontology $\mathcal{T}$, *i.e.*, $\mathcal{T} \not\models S \sqcap R \sqsubseteq \bot$.
- *Subsumption*: checks if S is more specific than R w.r.t. the ontology $\mathcal{T}$, *i.e.*, $\mathcal{T} \models S \sqsubseteq R$.

In real-world application scenarios featuring articulated and –oftentimes– conflicting descriptions, standard inference services like *Subsumption* and *Satisfiability* are inadequate, as they provide just a Boolean answer. An outcome explanation is required in more advanced settings, dealing with heterogeneous information from several independent sources. Non-monotonic reasoning tasks originally defined for belief revision are needed. The following non-standard inferences are particularly relevant for the purposes of this work:

- *Concept Contraction* [104]: if $\mathcal{T} \models S \sqcap R \sqsubseteq \bot$, *i.e.*, S and R are not compatible with each other, Concept Contraction (CC) is able to determine a pair of concepts $\langle G, K \rangle$ such that $\mathcal{T} \models R \equiv G \sqcap K$, and $K \sqcap S$ is satisfiable in $\mathcal{T}$. Then K is called a *contraction* of R according to S and $\mathcal{T}$. G (for *Give up*) represents "why" R is incompatible with S *i.e.*, which part of R is conflicting with S and must be retracted to obtain an expression K (for *Keep*) such that $K \sqcap S$ is satisfiable in $\mathcal{T}$. Hence, Concept Contraction Problem (CCP) amounts to an extension of (in)satisfiability.

- *Concept Abduction* [104]: if $\mathcal{T} \models S \sqcap R \not\sqsubseteq \bot$ and $\mathcal{T} \models S \not\sqsubseteq R$, *i.e.*, S and R are compatible but S does not subsume R, then Concept Abduction (CA) finds a concept H (for *Hypothesis*) such that $\mathcal{T} \models S \sqcap H \sqsubseteq R$. Basically, H represents what is in R but not specified in S. In particular, solving a Concept Abduction Problem (CAP) provides an extension and an explanation to (missed) subsumption.

- *Concept Bonus* [38]: extracts a concept B from S which denotes what S provides even though it is not specified in R. Formally, finding the Bonus B of S with respect to R is equivalent to find H in a Concept Abduction problem where R and S are swapped.

For both Concept Abduction and Concept Contraction , minimality criteria are defined –since one usually wants to hypothesize or give up as little as possible– which induce numerical distance (*penalty*) functions, based on the CNF norm of expressions H and G respectively. These penalties can be combined through a *utility function* (*a.k.a. score combination function*) in order

to evaluate goodness of match approximation and are studied and applied in *semantic matchmaking* problems [38]. Basically, semantic matchmaking is the process of finding the best matches to a *request* (named R in the above definitions) among available *resource* descriptions (named S above), where both the request and the resources are semantically annotated w.r.t. a common reference ontology [38]. In such scenarios, the result is a ranked list of appealing resources with respect to the request.

The principles underpinning semantic matchmaking are:

- *Open World Assumption.* The absence of a characteristic in a description should not be interpreted as a constraint of absence, but as unknown or irrelevant information.

- *Non-symmetric evaluation.* A semantic matchmaking system may give different evaluations to the match between two concept expressions S and R, depending on whether it is trying to match S with R, or R with S.

Five different match classes (categories) [47, 77], summarized in Table 2.3, are originated by matchmaking process. The most desired match is obviously the exact one, but from the viewpoint of a requester full match is equally acceptable. However, potential and partial matches are the most common in real complex scenarios. By means of Concept Contraction and Concept Abduction it is possible to move from a partial match to a full one by exploiting a query refinement process:

$$\textbf{partial} \rightarrow \textbf{potential} \rightarrow \textbf{full}$$

2.2.3 State of the art

The widespread and stable availability of hefty computational and networking resources characterizes classical Semantic Web contexts. Conversely, in SWoT scenarios, hardware is severely constrained and information sources are distributed across physical environments on tiny devices, inducing several issues. Highly volatile connectivity, unexpected disconnections, location dependency, and limitations of wireless communication links and energy supply make pervasive scenarios different from traditional wired and dependable

Table 2.3: Match classes

Match class	**Description**	**Semantics**
Exact	S ia semantically equivalent to R. All the requirements expressed in R are in S and S does not expose any additional feature w.r.t. R.	$\mathcal{T} \models R \equiv S$
Full	S is more specific than R. All the requirements expressed in R are provided by S and S exposes further characteristics both not required by R and not in conflict with the ones in R.	$\mathcal{T} \models S \sqsubseteq R$
Plug-in	R is more specific than S. All the characteristics expressed in S are requested by R and R exposes also other requirements both not exposed by S and not in conflict with characteristics in S.	$\mathcal{T} \models R \sqsubseteq S$
Potential	R is compatible with S. Nothing in R is logically in conflict with anything in S and vice-versa.	$\mathcal{T} \not\models S \sqcap R \sqsubseteq \bot$
Partial	R is not compatible with S. At least one requirement in R is logically in conflict with some characteristic in S.	$\mathcal{T} \models S \sqcap R \sqsubseteq \bot$

computing contexts. Mobile agents endowed with quick decision support, query answering and stream reasoning capabilities are required [55, 81] to support user activities and provide general-purpose innovative services. In these contexts, reasoning engines are exploited as decisional and organizational systems: specific non-standard inference services may be more suitable than standard ones [114]. Additionally, since mobile and embedded devices typically have limited resources, they require inference engines optimized to run efficiently on low-resource platforms, whereas common reasoners typically impose non-trivial hardware and software constraints, particularly on main memory.

Due to architectural constraints and computational complexity of Description Logics reasoning, the majority of early mobile inference engines provided only rule processing for entailment materialization in a KB. Proposals include *3APL-M* [75], *COROR* [127], *MiRE4OWL* [72], *Delta-Reasoner* [92] and the system in [115], that results unsuitable to support applications requiring non-standard inference tasks and extensive reasoning over ontologies [92].

Pocket KRHyper [123], a Java Micro Edition library for theorem proving and model generation based on the hyper tableau calculus, was the first reasoning engine specifically designed for mobile devices. It was exploited in a DL-based matchmaking framework between user profiles and descriptions of mobile resources/services [74]. However, frequent "out of memory" errors strongly limited the size and complexity of manageable logic expressions.

To overcome these constraints, tableaux optimizations to reduce memory consumption were introduced in [126] and implemented in *mTableaux*, a modified version of Java Standard Edition (SE) *Pellet* reasoner [124]. Comparative performance tests were performed on a PC, showing faster turnaround times than both unmodified Pellet and *Racer* [64] reasoners. Nevertheless, the Java SE technology is not tailored to mobile and embedded devices. In fact, several inference engines cannot run on common mobile platforms, since they rely on Java class libraries incompatible with most widespread mobile OS (*e.g.*, Android). In [142] four Semantic Web reasoners were successfully ported to the Android platform (Pellet, *CB* [69], *Hermit* [120] and *JFact*, a Java port of *Fact++* [131]), albeit with significant rewriting or restructuring

effort in some cases. Similarly, in [70] the *ELK* reasoner was optimized and evaluated on Android. However, all ported systems were designed mainly for batch jobs over large ontologies and/or expressive languages. This makes mobile device usage less suitable due to computation and memory constraints.

Furthermore, the above reasoners only support standard inference services such as satisfiability and subsumption, which are not enough for pervasive scenarios. Non-monotonic reasoning tasks are needed in order to enable the creation of software agents able to provide quick decision support and/or on-the-fly organization in such intrinsically unpredictable environments.

Mini-ME [113] was an Android-oriented *matchmaker* and reasoner, compatible with Java SE. It provides both standard and non-standard inferences for semantic matchmaking. The rule engine for Android in [138] introduced the novel *RETEpool* algorithm on OWL 2 RL rulesets, capable of balancing memory usage and time performance. *Mini-ME Swift* [107] was the first OWL reasoner for iOS, re-designed from the above Mini-ME with the OWL API for iOS [106].

Tiny-ME (the Tiny Matchmaking Engine) [105] is a matchmaking and reasoning engine designed and implemented with a compact and portable C core. It provides a larger set of standard and non-standard inference services w.r.t. its predecessor Mini-ME in a moderately expressive OWL 2 fragment corresponding to the $\mathcal{ALN}$ DL. It has a multiplatform architecture with multiple Applications Programming Interfaces (APIs) ensuring native support for Windows, Linux, macOS, Docker containers, Android, iOS, and embedded systems like Apache NuttX. In this work, the Tiny-ME C version has been integrated into the deductive argumentative reasoning engine prototype discussed in Chapter 4.

2.3 Argumentation basics

For people, argumentation is a pervasive phenomenon in everyday life; it is particularly important for negotiating decisions and managing conflicts, subjective points of view, opinions, objectives, and in general any scenario in which the available information is incomplete or inconsistent. Several theoretical studies have attempted to formalize the dynamics that characterize it.

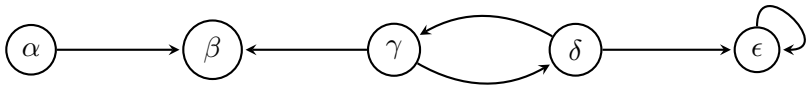


Figure 2.4: $\mathcal{F}_1$: AF example

The interest in these issues has pushed the scientific community towards the creation of computational tools and models in areas such as human-machine dialogue systems, multi-agent systems, decision support systems and planning: over the last years, computational argumentation has been gaining increasing relevance across the board within Artificial Intelligence (AI) [19]. *Argumentation* is the process by which arguments and counterarguments are built and assessed. It may involve comparing arguments, evaluating them in some respects, and judging a constellation of arguments and counterarguments to consider whether any subset of them is warranted according to some criterion [23]. *Abstract Argumentation* (AA) [49] assesses the acceptability of each argument solely in terms of those relations, abstracted from the content of the arguments themselves. In contrast, *Structured Argumentation* (SA) [21] uses formal models to represent arguments and relies on inferences to evaluate relationships.

Dung's seminal work [49] defined an *Argumentation Framework* (AF) as a graph-based formalism to reason over conflicting knowledge without considering the internal structure of the arguments, but only their mutual relations of *attack* –denoting the conflicts between pairs of arguments– and the *semantics* for evaluating them, *i.e.*, for determining what arguments can be considered as acceptable.

Definition 1 (Argumentation Framework) *An Argumentation Framework (AF) is a pair $\mathcal{F} = \langle \mathcal{A}, \mathcal{R} \rangle$, where $\mathcal{A}$ is a finite set of arguments and $\mathcal{R} \subseteq \mathcal{A} \times \mathcal{A}$ is the set of relations. The relation $\alpha\ \mathcal{R}\ \beta$ means that α attacks β, or equivalently β is attacked by α.*

Commonly, an AF is represented as a directed graph whose nodes are the arguments and each edge between a pair of nodes represents the relation from the attacker to the attacked argument. An example of AF is depicted in Figure 2.4.

2.3.1 Argumentation semantics

An argumentation semantics is the formal definition of a method ruling the argument evaluation process. In order to evaluate the justification of an argument the notion of *defense* is important.

Definition 2 (Defense) *Given an AF* $\mathcal{F} = \langle \mathcal{A}, \mathcal{R} \rangle$*, a set* $\mathcal{S} \subseteq \mathcal{A}$ *and an argument* $\alpha \in \mathcal{A}$*,* α *is defended by* $\mathcal{S}$ *iff for each* $\beta \in \mathcal{A}$*:* $\beta \ \mathcal{R} \ \alpha \ \Rightarrow \ \exists \ \gamma \in \mathcal{S} \mid \gamma \ \mathcal{R} \ \beta$*.*

Extension-based semantics

In the *extension-based* approach [17], a semantics specifies how to derive from an AF a set of extensions, where an *extension* E of an AF $\langle \mathcal{A}, \mathcal{R} \rangle$ is a subset of $\mathcal{A}$, intuitively representing a set of arguments which can "survive together" or are "collectively acceptable". Thus, a specific extension-based argumentation semantics provides a way to select reasonable sets of arguments among all the possible ones, according to some criterion embedded in its definition.

Definition 3 (Extension-based Semantics) *Given an AF* $\mathcal{F} = \langle \mathcal{A}, \mathcal{R} \rangle$*, an extension-based semantics* $\mathcal{S}$ *associates* $\mathcal{F}$ *with a subset of* $2^{\mathcal{A}}$*, denoted as* $\mathcal{E}_{\mathcal{S}}(\mathcal{F})$*.*

This approach enables the existence of a set of alternatives for a single argumentation framework. However, it can happen that a semantics is defined so that a single outcome is prescribed for each AF. Formally, if for any AF $\mathcal{F}$ it holds that $|\mathcal{E}_{\mathcal{S}}(\mathcal{F})| = 1$ then the semantics is said to belong to the *unique-status* (or single-status) approach, while in general it is said to belong to the *multiple-status* approach. Furthermore, one or many extensions may be prescribed by a given semantics $\mathcal{S}$ but in general, it is possible that no extensions are prescribed for an AF, *i.e.*, $\mathcal{E}_{\mathcal{S}}(\mathcal{F}) = \varnothing$. This corresponds to the case where the semantics $\mathcal{S}$ is undefined in $\mathcal{F}$ since no extensions compliant with the definition of $\mathcal{S}$ exist. In the following, $\mathcal{D}_{\mathcal{S}}$ denotes the set of AFs where a semantics $\mathcal{S}$ is defined, *i.e.*, $\mathcal{D}_{\mathcal{S}} = \{\mathcal{F} \mid \mathcal{E}_{\mathcal{S}}(\mathcal{F}) \neq \varnothing\}$.

The most basic concept shared by all argumentation semantics in literature is *conflict-freeness*. Arguments in an extension should not be in conflict

with each other. Consequently, if an argument α attacks another argument β, then α and β can not be in the same extension.

Definition 4 (Conflict-freeness) *Given an AF $\mathcal{F} = \langle \mathcal{A}, \mathcal{R} \rangle$, a set of arguments $\mathcal{S} \subseteq \mathcal{A}$ is conflict-free iff there are not $\alpha, \beta \in \mathcal{S}$, such that $\alpha \ \mathcal{R} \ \beta$. The collection of all conflict-free is denoted as $cf(\mathcal{F})$.*

By this definition, self-attacking arguments can never be in any conflict-free set. As all semantics share the concept of conflict-freeness, self-attacking arguments cannot be part of any extension of any semantics. Since $\varnothing$ is conflict-free by definition, there is always at least one conflict-free set in any arbitrary AF.

Example 1 *Consider the AF $\mathcal{F}_1 = \langle \mathcal{A}_1, \mathcal{R}_1 \rangle$ depicted in Figure 2.4. The conflict-free sets in $\mathcal{F}_1$ are $cf(\mathcal{F}_1) = \{\varnothing, \{\alpha\}, \{\beta\}, \{\gamma\}, \{\delta\}, \{\alpha, \gamma\}, \{\alpha, \delta\}, \{\beta, \delta\}\}$. Since ϵ is self-attacking, there is no $E \in cf(\mathcal{F}_1)$ with $\epsilon \in E$.*

A further requirement corresponds to the idea that an extension is a set of arguments which "can stand on its own", *i.e.*, it is able to withstand the attacks it receives from other arguments by replying with other attacks. Formally, this corresponds to the property of *admissibility*.

Definition 5 (Admissible set) *Given an AF $\mathcal{F} = \langle \mathcal{A}, \mathcal{R} \rangle$, a set of arguments $\mathcal{S} \subseteq \mathcal{A}$ is an admissible set if $\mathcal{S}$ is conflict-free and $\mathcal{S}$ is defended by itself,* i.e., *$\forall \ \beta \in \mathcal{A}$, $\forall \ \alpha \in \mathcal{S}$, $\beta \ \mathcal{R} \ \alpha \Rightarrow \exists \ \gamma \in \mathcal{S}$ such that $\gamma \ \mathcal{R} \ \beta$. The collection of all admissible sets is denoted as $adm(\mathcal{F})$.*

By definition, there is always at least one admissible set, as $\varnothing$ is not only conflict-free, but also defended by itself, and therefore $\varnothing \in adm(F)$ for any AF $\mathcal{F}$.

Example 2 *Consider the AF $\mathcal{F}_1 = \langle \mathcal{A}_1, \mathcal{R}_1 \rangle$ in Figure 2.4. The admissible sets are $adm(\mathcal{F}_1) = \{\varnothing, \{\alpha\}, \{\gamma\}, \{\delta\}, \{\alpha, \gamma\}, \{\alpha, \delta\}\}$. Considering the conflict-free set $\{\beta, \delta\}$, argument δ defends itself and β from attacker γ, but β is not defended from attacker α. Therefore $\{\beta, \delta\}$ is not admissible. On the other hand, the set $\{\gamma, \epsilon\}$ is defended by itself but not conflict-free as ϵ attacks itself, therefore it is not admissible either.*

Definition 6 (Admissibility Principle) *Given an AF* $\mathcal{F} = \langle \mathcal{A}, \mathcal{R} \rangle$*, a semantics* $\mathcal{S}$ *satisfies the admissibility principle iff* $\forall\ \mathcal{F} \in \mathcal{D}_{\mathcal{S}}$*:* $\mathcal{E}_{\mathcal{S}}(\mathcal{F}) \subseteq adm(\mathcal{F})$, i.e., $\forall\ E \in \mathcal{E}_{\mathcal{S}}(\mathcal{F})$ *it holds that:*

$$\alpha \in E \Rightarrow (\forall\ \beta \in \mathcal{A}\text{:}\ \beta\ \mathcal{R}\ \alpha \Rightarrow E\ \mathcal{R}\ \beta).$$

The property of *reinstatement* is an essential feature of defeasible reasoning w.r.t. the notion of defense. Intuitively, if the attackers of an argument α are in turn attacked by an extension E, it may be assumed they have no effect on α: then α should be, in a sense, reinstated, therefore it should belong to E. This leads to the following reinstatement principle.

Definition 7 (Reinstatement Principle) *A semantics* $\mathcal{S}$ *satisfies the reinstatement principle iff* $\forall\ \mathcal{F} \in \mathcal{D}_{\mathcal{S}}$, $\forall\ E \in \mathcal{E}_{\mathcal{S}}$ *it holds that:*

$$(\forall\ \beta \in \mathcal{A}\text{:}\ \beta\ \mathcal{R}\ \alpha \Rightarrow E\ \mathcal{R}\ \beta) \Rightarrow \alpha \in E.$$

The justification state of an argument α can be conceived in terms of its extension membership. A basic classification encompasses only two possible states for an argument, either justified or not justified. In this respect, two alternative types of justification can be considered.

Definition 8 (Types of justification) *Given a semantics* $\mathcal{S}$ *and an AF* $\mathcal{F} \in \mathcal{D}_{\mathcal{S}}$*, an argument* α *is:*

- *skeptically justified iff* $\forall\ E \in \mathcal{E}_{\mathcal{S}}(\mathcal{F})$*:* $\alpha \in E$*;*
- *credulously justified iff* $\exists\ E \in \mathcal{E}_{\mathcal{S}}(\mathcal{F})$*:* $\alpha \in E$*.*

Clearly, the two notions coincide for unique-status approaches, while, in general, credulous justification includes skeptical justification.

Adopting the aforementioned general notions, several extension-based semantics have been proposed in literature. From a historical point of view, it is possible to distinguish between:

- four "traditional" semantics, introduced in Dung's original paper [49] and called *complete*, *grounded*, *preferred*, and *stable*;

- subsequent proposals introduced by various authors, often to overcome limitations or fix undesired behaviors of traditional approaches; they include *naive* [26], *stage* [133], *semi-stable* [33], *ideal* [50], *eager* [34], *cf2* [16], *stage2* [53], *resolution-based grounded* [15], and *prudent* [40] semantics.

From a taxonomy point of view, semantics can be classified into admissible- and naive-based. *Admissible-based semantics* are those semantics which are built on the concept of admissible sets. *Naive-based semantics* are those semantics which are built to overcome the requirement of accepting arguments outside an odd-length attack cycle. On the basis of this requirement, all Dung' semantics fall into the category of admissible-based semantics, whereas naive, stage, cf2 and stage2 belong to the naive-based semantics.

Ranking semantics

In applications involving a large number of arguments, it can be problematic to have only two levels of evaluation, *i.e.*, arguments either accepted or rejected. For instance, this is a limitation when using argumentation for decision support systems in pervasive contexts, where the collected information is highly heterogeneous, conflicts are frequent and even sub-optimal solutions can be considered useful, due to unpredictable context changes and strict computational limitations of devices. In order to overcome these shortcomings, one may want to adopt semantics which distinguish arguments with a larger number of degrees of acceptability; the degree of acceptability of an argument is sometimes called *overall strength*. A possible approach to select a set of acceptable arguments is to rank them from the most to the least acceptable. *Ranking-based* semantics [7, 28] (*a.k.a. gradual semantics*) aim at determining such an ordering in among arguments. Several ranking-based semantics can be found in literature, with a range of behaviors and logical properties.

Definition 9 (Ranking-based Semantics) *A ranking-based semantics $\mathcal{S}$ associates to any AF $\mathcal{F} = \langle \mathcal{A}, \mathcal{R} \rangle$ a ranking $\succeq_{\mathcal{F}}^{\mathcal{S}}$ on $\mathcal{A}$, where $\succeq_{\mathcal{F}}^{\mathcal{S}}$ is a preorder (*i.e.*, a reflexive and transitive relation) on $\mathcal{A}$. $\alpha \succeq_{\mathcal{F}}^{\mathcal{S}} \beta$ means that α is at least as acceptable as β.*

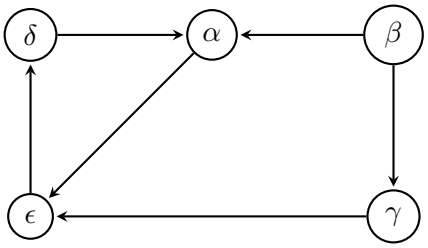


Figure 2.5: $\mathcal{F}_2$: AF example for Ranking-based semantics

Useful notions are recalled in what follows, in order to formalize ranking-based semantics as required in this work.

Definition 10 *Let $\mathcal{F} = \langle \mathcal{A}, \mathcal{R} \rangle$ be an AF and $\alpha, \beta \in \mathcal{A}$. Then,*

- *a **path** p from α to β is a sequence $\langle a_0, \ldots, a_n \rangle$ of arguments such that $a_0 = \alpha$, $a_n = \beta$ and $\forall i < n, (a_i, a_{i+1}) \in \mathcal{R}$. The **length** l_p of the path p is n;*

- *a **defender** of β is an argument at the beginning of an even-length path. The multiset of defenders of β is denoted by $\mathcal{R}_n^+(\beta)$. An argument β is defended if $\mathcal{R}_2^+(\beta) \neq \varnothing$;*

- *an **attacker** of β is an argument at the beginning of an odd-length path. The multiset of attackers is denoted by $\mathcal{R}_n^-(\beta)$. The **direct attackers** of an argument β are the arguments in $\mathcal{R}_1^-(\beta)$.*

In literature ranking-based semantics have been characterized by means of several logical properties, including *abstraction* (Abs), *independence* (In), *void precedence* (VP), *self-contradiction* (SC), *cardinality precedence* (CP), *quality precedence* (QP), *counter transitivity* (CT), *defense precedence* (DP); their definitions can be found in [28]. All these properties are not mandatory for a single semantics, but they are useful as informative indicators to highlight the differences in behavior between them. Some well-known ranking-based semantics that return a unique ranking between arguments are recalled hereafter.

The work in [22] has proposed a *Categoriser* function, which assigns a score to each argument, given the value of its direct attackers.

Definition 11 (Categoriser) *Given an AF* $\mathcal{F} = \langle \mathcal{A}, \mathcal{R} \rangle$*, the categoriser function* $Cat : \mathcal{A} \mapsto]0, 1]$ *is defined as:*

$$Cat(\alpha) = \begin{cases} 1 & if\ \mathcal{R}_1^-(\alpha) = \varnothing \\ \frac{1}{1+\sum_{\gamma \in \mathcal{R}_1^-(\alpha)} Cat(\gamma)} & otherwise \end{cases}$$

The ranking-based semantics Categoriser associates to $\mathcal{F}$ *a ranking* $\succeq_{\mathcal{F}}^{Cat}$ *on* $\mathcal{A}$ *such that* $\forall\ \alpha, \beta \in \mathcal{A}, \alpha \succeq_{\mathcal{F}}^{Cat} \beta$ *iff* $Cat(\alpha) \geq Cat(\beta)$*.*

Example 3 *Consider the AF* $\mathcal{F}_2$ *depicted in Figure 2.5. The categorisers are* $Cat(\alpha) \approx 0.38$*,* $Cat(\beta) \approx 1$*,* $Cat(\gamma) \approx 0.5$*,* $Cat(\delta) \approx 0.65$*,* $Cat(\epsilon) \approx 0.53$*. Therefore, the ranking is* $\beta \succ_{\mathcal{F}_2}^{Cat} \delta \succ_{\mathcal{F}_2}^{Cat} \epsilon \succ_{\mathcal{F}_2}^{Cat} \gamma \succ_{\mathcal{F}_2}^{Cat} \alpha$*.*

Categoriser takes into account only the value of the direct attackers to compute the overall strength of an argument. In fact, in $\mathcal{F}_2$ the argument ϵ, which is attacked by arguments that are attacked by a non-attacked argument, is ranked higher than γ, which is attacked just once, but by a stronger argument.

The *Discussion*-based semantics (`Dbs`) [7] is centered on the notion of *linear discussion*, which recalls the "argumentation line" defined in [57]. A linear discussion is a sequence of arguments such that each argument attacks the argument preceding it in the sequence.

Definition 12 (Linear discussion) *Given an AF* $\mathcal{F} = \langle \mathcal{A}, \mathcal{R} \rangle$ *and an argument* $\alpha \in \mathcal{A}$*, a linear discussion for* α *in* $\mathcal{F}$ *is a sequence* $s = \langle a_1, \ldots, a_n \rangle$ *of elements of* $\mathcal{A}$ *such that* $a_1 = \alpha$ *and* $\forall i \in \{2, 3, \ldots, n\}\ a_i\ \mathcal{R}\ a_{i-1}$*. The length of the linear discussion* s *is* n*;* s *is* ***won*** *iff* n *is odd. Conversely,* s *is* ***lost*** *iff* n *is even.*

`Dbs` compares arguments by counting, for every positive integer i, the number of linear discussions of length i for every argument. Lost discussions are counted positively while won discussions are counted negatively, so that the smaller the calculated number, the better the outcome for the reference argument.

Definition 13 (Discussion-based Semantics) *Given an AF* $\mathcal{F} = \langle \mathcal{A}, \mathcal{R} \rangle$*,* $\alpha \in \mathcal{A}$ *and* $i \in \mathbb{N}^+$*.*

The discussion count of α, denoted $Dis_{\mathcal{F}}(\alpha) = \langle Dis_{\mathcal{F}_1}(\alpha), Dis_{\mathcal{F}_2}(\alpha), \cdots \rangle$, is defined as:

$$Dis_{\mathcal{F}_i}(\alpha) = \begin{cases} -N & if\ i\ is\ odd \\ N & if\ i\ is\ even \end{cases}$$

where N is the number of linear discussions for α in $\mathcal{F}$ of length i.

The ranking-based semantics `Dbs` *associates to all argumments in an AF $\mathcal{F}$, taken by lexicographical order, a ranking $\succeq_{\mathcal{F}}^{Dbs}$ on $\mathcal{A}$ such that $\forall\ \alpha, \beta \in \mathcal{A}, \alpha \succeq_{\mathcal{F}}^{Dbs} \beta$ iff one of the two following cases holds:*

- $\forall i \in \{1, 2, \dots\}$, $Dis_{\mathcal{F}_i}(\alpha) = Dis_{\mathcal{F}_i}(\beta)$;
- $\exists\ i \in \{1, 2, \dots\}$, $Dis_{\mathcal{F}_i}(\alpha) < Dis_{\mathcal{F}_i}(\beta)$ *and* $\forall\ j \in \{1, 2, \dots, i-1\}$, $Dis_{\mathcal{F}_j}(\alpha) = Dis_{\mathcal{F}_j}(\beta)$.

Table 2.4: Count of discussions for the arguments in $\mathcal{F}_2$

i	α	β	γ	δ	ϵ
1	-1	-1	-1	-1	-1
2	2	0	1	1	2
3	-1	0	0	-2	-3

Example 4 *Consider the AF $\mathcal{F}_2$ depicted in Figure 2.5 and the Table 2.4 of linear discussions number for every argument in $\mathcal{F}_2$. By adopting* `Dbs`*, the obtained ranking is $\beta \succ_{\mathcal{F}_2}^{Dbs} \delta \succ_{\mathcal{F}_2}^{Dbs} \gamma \succ_{\mathcal{F}_2}^{Dbs} \epsilon \succ_{\mathcal{F}_2}^{Dbs} \alpha$.*

Another well-known semantics is the *Burden-based semantics* (`Bbs`) [7]. It follows a multi-step process, where a *burden* number is assigned to every argument at each step. In the initial step, this number is 1 for all arguments. Then, in each step, all the burden numbers are simultaneously recomputed on the basis of the number of direct attackers and their burden numbers from the previous step.

Definition 14 (Burden-based Semantics) *Given an AF $\mathcal{F} = \langle \mathcal{A}, \mathcal{R} \rangle$, $\alpha \in \mathcal{A}$ and $i \in \mathbb{N}$.*

The Burden number of α*, denoted by* $Bur_{\mathcal{F}}(\alpha) = \langle Bur_{\mathcal{F}_0}(\alpha), Bur_{\mathcal{F}_1}(\alpha), \cdots \rangle$*, is defined as:*

$$Bur_{\mathcal{F}_i}(\alpha) = \begin{cases} 1 & if\ i\ =\ 0 \\ 1 + \sum_{\beta \in \mathcal{R}_1^-(\alpha)} \frac{1}{Bur_{\mathcal{F}_{i-1}}(\beta)} & otherwise \end{cases}$$

where by convention if $\mathcal{R}_1^-(\alpha) = \varnothing$*, then* $\sum_{\beta \in R_1^-(\alpha)} \frac{1}{Bur_{\mathcal{F}_{i-1}}(\beta)} = 0$*.*

The ranking-based semantics **Bbs** *associates to any AF* $\mathcal{F}$ *by using the lexicographical order, a ranking* $\succeq_{\mathcal{F}}^{Bbs}$ *on* $\mathcal{A}$ *such that* $\forall \alpha, \beta \in \mathcal{A}, \alpha \succeq_{\mathcal{F}}^{Bbs} \beta$ *iff one of the two following cases holds:*

- $\forall i \in \{0, 1, \dots\}$, $Bur_{\mathcal{F}_i}(\alpha) = Bur_{\mathcal{F}_i}(\beta)$*;*
- $\exists i \in \{0, 1, \dots\}$, $Bur_{\mathcal{F}_i}(\alpha) < Bur_{\mathcal{F}_i}(\beta)$ *and* $\forall j \in \{0, 1, \dots, i-1\}$, $Bur_{\mathcal{F}_j}(\alpha) = Bur_{\mathcal{F}_j}(\beta)$*.*

Table 2.5: Burden numbers for the arguments in $\mathcal{F}_2$

i	α	β	γ	δ	ϵ
0	1	1	1	1	1
1	3	1	2	2	3
2	2.5	1	2	1.33	1.83

Example 5 *Consider the AF* $\mathcal{F}_2$ *depicted in Figure 2.5 and the Table 2.5 which presents the burden numbers for every argument in* $\mathcal{F}_2$*. By adopting* **Bbs***, the ranking obtained is* $\beta \succ_{\mathcal{F}_2}^{Bbs} \delta \succ_{\mathcal{F}_2}^{Bbs} \gamma \succ_{\mathcal{F}_2}^{Bbs} \epsilon \succ_{\mathcal{F}_2}^{Bbs} \alpha$*.*

Like in the previous two examples, `Dbs` and `Bbs` often return the same ranking, since both semantics disregard possible dependencies between an argument and one of its attackers, as well as the dependencies between two attackers. Also, `Dbs` and `Bbs` are not able to handle AFs with attack cycles between arguments; they produce the ranking only on the basis of the structure obtained by "unrolling" the cycles.

2.3.2 Generalizations of Argumentation Framework

Standard argumentation frameworks provide various semantics for acceptability, but there are still several limitations. There have been a number of

proposals for extending Dung's framework in order to allow for more sophisticated modeling and analysis of conflicting information. A common idea in some proposals is the observation that not all arguments and their relations are equal: in many scenarios there exists a propensity to define preference relations or to add values/weights to the arguments. Furthermore, in Dung's AF it is only possible to represent support implicitly, via defended arguments, but in many situations an independent type of relations w.r.t. attacks could be useful. Moreover, all attacks have always the same strength, but it might be beneficial to assign different levels of strength for attacks, *e.g.*, gradual weights associated with attacks. In order to cope with the above requirements, several generalizations of the basic AF have been proposed. Hereafter, the most relevant ones for this work are briefly recalled.

Bipolar Argumentation Framework

Generally speaking, pairs of arguments may be in conflict, due *e.g.*, to the presence of inconsistency in knowledge bases. Indeed, in all argumentation systems, an attack relation is considered in order to capture the conflicts. However, most logical theories of argumentation assume that defense is an implicit form of support. For instance, if an argument α attacks an argument γ and γ attacks an argument β, then α supports β. In this case, the notion of support does not have to be formalized differently from the notion of attack and Dung's AF model is enough, in which only one kind of interaction is explicitly represented by the attack relation. In this context, the support of β by another argument α can be represented only if α *defends* β in the meaning of [41]. It is a parsimonious strategy, but it is not a correct description of the process of argumentation in general.

A *Bipolar Argumentation Framework* (BAF) [36, 10] is an extension of the standard AF, in which two kinds of interactions between arguments are possible: attack and support. These two relations are independent, which leads to a bipolar representation of the interplay between arguments. Hence, the BAF can convey explicitly the difference between defense and support.

Definition 15 (Bipolar AF) *A BAF is a triplet* $\mathcal{B} = \langle \mathcal{A}, \mathcal{R}_{att}, \mathcal{R}_{sup} \rangle$*, where* $\mathcal{A}$ *is a finite set of arguments,* $\mathcal{R}_{att}$ *is a binary relation on* $\mathcal{A}$ *called attack*

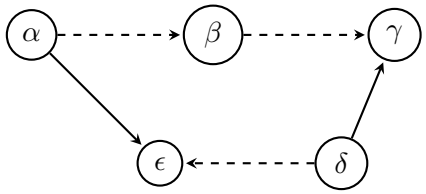


Figure 2.6: BAF example

and $\mathcal{R}_{sup}$ is a binary relation on $\mathcal{A}$ called support. Given two arguments α and β, α $\mathcal{R}$_attβ (respectively α $\mathcal{R}_{sup}$ β) means that α attacks β (resp. α supports β).

A BAF is typically represented by a directed graph, where two kinds of edges are used to distinguish the two relation types: solid edges denote attacks from the attacker to the attacked node, while dashed edges denote supports. A BAF example is depicted in Figure 2.6.

In BAFs, new kinds of attack emerge from the interaction between the direct attacks and the supports: there are a *supported attack*, represented by a sequence of supports followed by one attack, and an *indirect attack*, represented by an attack followed by a sequence of supports. Formally:

Definition 16 (BAF Supported and Indirect Attacks) *Let $\mathcal{B} = \langle \mathcal{A}, \mathcal{R}_{att}, \mathcal{R}_{sup} \rangle$ be a BAF.*

*A **supported attack** for an argument $\beta \in \mathcal{A}$ from an argument α_0 is a sequence α_0 $\mathcal{R}_1$ $\cdots$ $\mathcal{R}_n$ β, with $n \geq 2$, such that $\mathcal{R}_i = \mathcal{R}_{sup}$ $\forall$ $i = 1, \ldots, n-1$, and $\mathcal{R}_n = \mathcal{R}_{att}$.*

*An **indirect attack** for an argument $\beta \in \mathcal{A}$ from an argument α_0 is a sequence α_0 $\mathcal{R}_1$ $\cdots$ $\mathcal{R}_n$ β, with $n \geq 2$, such that $\mathcal{R}_i = \mathcal{R}_{sup}$ $\forall$ $i = 2, \ldots, n$, and $\mathcal{R}_1 = \mathcal{R}_{att}$.*

Taking into account sequences of supports and attacks leads to the following definitions, applicable to sets of arguments.

Definition 17 (Defense for BAF) *Let $\mathcal{B} = \langle \mathcal{A}, \mathcal{R}_{att}, \mathcal{R}_{sup} \rangle$ be a BAF.*

*A set $\mathcal{S} \subseteq \mathcal{A}$ **set-attacks** an argument $\beta \in \mathcal{A}$, iff there exists a supported attack or an indirect attack from an element of $\mathcal{S}$ to β.*

A set $\mathcal{S} \subseteq \mathcal{A}$ ***set-supports*** *an argument* $\beta \in \mathcal{A}$*, iff there exists a sequence* $\alpha_1 \ \mathcal{R}_{sup} \ \ldots \ \mathcal{R}_{sup} \ \alpha_n$*, with* $n \geq 2$*, such that* $\alpha_n = \beta$ *and* $\alpha_1 \in \mathcal{S}$*.*

A set $\mathcal{S} \subseteq \mathcal{A}$ ***defends*** *collectively an argument* $\alpha \in \mathcal{A}$*, iff for each argument* $\beta \in \mathcal{A}$*, if* $\{\beta\}$ *set-attacks* α*, then* $\exists \ \gamma \in \mathcal{S}$ *such that* $\{\gamma\}$ *set-attacks* β*.*

The acceptability of arguments has been investigated also in the case of bipolar framework. Following Dung's methodology, in the original BAF proposal [36] the well-known stable and preferred semantics are generalized for a BAF by enforcing the coherence requirement for an acceptable set of arguments. A more recent work [140] provides a principle-based analysis of 28 BAF semantics to obtain a better understanding of the different kinds of support (*i.e.*, deductive, necessary, evidential) and to choose the best-fit argumentation semantics to a particular application.

Weighted Argumentation Framework

A *Weighted Argumentation Framework* (WAF) [52, 51] is an extension of the standard AF, in which attacks between arguments are associated with a numeric weight, indicating the relative strength of the attack. There can be several interpretations of these weights:

- *Weighted Majority Relations*: represents a weight as the number of votes in support of the attack, in the context of collective decision-making. The most natural interpretation of the attack weights refers to the number of agents of a given group in accordance with the attack relationship under consideration. In this case a WAF can be considered as an aggregated view of the classical argumentation frameworks within a group of agents. This is a straightforward way to aggregate the points of view of the agents.

- *Weights as Beliefs:* equates weights to subjective beliefs, assigning value false to the attacked argument when the attacking argument is believed true. In structured argumentation, unlike Dung's classic approach, arguments have their own internal structure, often represented in some logical formalism. In this context, the notion of attack can

be refined and different attack strengths can be defined by determining "how much" an argument attacks another one. This leads for instance to evaluate how much the conclusion of an argument is inconsistent with the premises of another one, using an inconsistency measure. Since it does not depend on any additional information, but just on the structure of the arguments, this situation is commonly called "implicit strength".

- *Weights as Ranking:* perhaps the most intuitive interpretation, in which weights are used to rank the relative strength of attacks between arguments, *i.e.*, the higher the weight, the stronger the attack. In this interpretation, the criteria for ranking attacks can be subjective or objective. Weights can reflect the evaluation of the strengths of the attacks between arguments. In some applications, it proves sensible to split the attack relation into a relatively small set of different types of attacks. A classic partition consists in labeling attacks either as weak, medium or strong: weak attacks are attacks that are not completely reliable, while strong ones are attacks that cannot be ignored. Different attack strengths can also be derived when attacks come from different sources, which can be more or less reliable, reputable or significant. This case is called "explicit strength", since it requires some additional information which must be given explicitly when the WAF is built.

Definition 18 (Weighted AF) *A WAF is a triplet $\mathcal{W} = \langle \mathcal{A}, \mathcal{R}, w \rangle$, where $\langle \mathcal{A}, \mathcal{R} \rangle$ is the standard AF and $w : \mathcal{R} \mapsto \mathbb{R}^+$ is a function assigning positive real-valued weights to attacks.*

As shown in Figure 2.7, a WAF can be represented by a directed graph, where the value labeling each edge represents the weight of the attack relation from the attacker to attacked node.

In this framework, attacks must have a strictly positive weight because allowing 0-weight attacks would be counter-intuitive, since it could be interpreted as the absence of attack relation. Here, the traditional notion of conflict-free sets of arguments is relaxed: some inconsistencies are tolerated in a set $\mathcal{S}$ of arguments, assuming that the sum of the weights of attacks between arguments in $\mathcal{S}$ does not exceed a given *inconsistency budget* $\beta \in \mathbb{R}^+$,

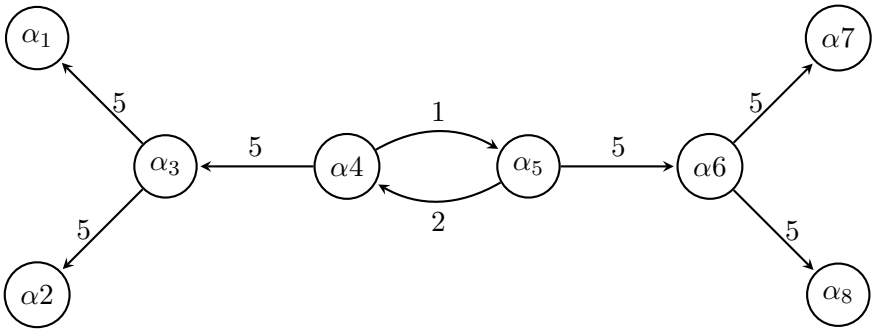


Figure 2.7: $\mathcal{W}_1$: WAF example

that is, given β, attacks up to a total weight of β could be neglected. Standard abstract argumentation systems assume an inconsistency budget of 0; by relaxing this constraint, WAFs can attain more solutions.

Definition 19 (Inconsistency Budget) *Let $\mathcal{W} = \langle \mathcal{A}, \mathcal{R}, w \rangle$ be a WAF. The function sub, which takes an attack relation $\mathcal{R}$, a weight function w, and an inconsistency budget $\beta \in \mathbb{R}^+$, returns the set of subsets $\mathcal{C}$ of $\mathcal{R}$ whose total weight does not exceed β,* i.e.*,*

$$sub(\mathcal{R}, w, \beta) = \{\mathcal{C} \subseteq \mathcal{R} \mid \sum_{\langle \alpha_1, \alpha_2 \rangle \in \mathcal{C}} w(\langle \alpha_1, \alpha_2 \rangle) \leq \beta\}$$

WAFs generalize unweighted AFs and have some advantages over them. As unweighted AFs give always consistent solutions, it may be the case that a consistent solution could be the empty set. WAFs, in contrast, have the following property: for every set of arguments $\mathcal{S} \subseteq \mathcal{A}$, Given a function f, $\mathcal{S}$ is β-f if $\exists\, \mathcal{C} \in sub(\mathcal{R}, w, \beta)$ such that $\mathcal{S} \in f(\langle \mathcal{A}, \mathcal{R} \backslash \mathcal{C} \rangle)$ [51]. For example, $\mathcal{S}$ is β-conflict free if $\exists\, \mathcal{C} \in sub(\mathcal{R}, w, \beta)$ such that $\mathcal{S}$ is conflict free in the argument system $\langle \mathcal{A}, \mathcal{R} \backslash \mathcal{C} \rangle$.

Example 6 *Consider the WAF $\mathcal{W}_1$ shown in Figure 2.7. The only conflict-free set of arguments in $\mathcal{W}_1$ is the empty set. However, for $\beta = 1$ the set $\{\alpha_5\}$ is 1-conflict-free, since the attack $\langle \alpha_4, \alpha_5 \rangle$ can be disregarded. For $\beta = 2$, the sets $\{\alpha_4\}$ and $\{\alpha_5\}$ are 2-conflict-free.*

Therefore in a WAF any set of arguments is conflict-free at some cost, and the cost required to make a set of arguments conflict-free immediately

Figure 2.8: BWAF legend: b_1 attacks a_1, b_2 supports a_2

provides a preference ordering over sets of arguments. Admissibility is defined in the standard way, and standard semantics are considered leading to various notions of β-extensions of Dung-style ones, *e.g.*, grounded, preferred, stable extensions [52].

Bipolar Weighted Argumentation Framework

As a further generalization of Dung-style AFs, the *Bipolar Weighted Argumentation Framework* (BWAF) [97] allows not only weighted attack relations between abstract arguments, but also weighted support relations. This is achieved by assigning a weight to each relation which can be positive or negative.

Definition 20 (Bipolar Weighted AF) *A BWAF is a triplet* $\mathcal{G} = \langle \mathcal{A}, \mathcal{R}, w \rangle$, *where* $\mathcal{A}$ *is a finite set of arguments,* $\mathcal{R} \subseteq \mathcal{A} \times \mathcal{A}$ *and* $w \colon \mathcal{R} \mapsto [-1, 0[\cup]0, 1]$. *Attack relations are defined as* $\mathcal{R}_{att} = \{\langle a, b \rangle \in \mathcal{R} \mid w(\langle a, b \rangle) \in [-1, 0[\ \}$ *and support relations as* $\mathcal{R}_{sup} = \{\langle a, b \rangle \in \mathcal{R} \mid w(\langle a, b \rangle) \in]0, 1] \ \}$. *Given two arguments* $a, b \in \mathcal{A}$ *and a path* $\langle a, x_1, x_2, \ldots, x_n, b \rangle$ *from* a *to* b, *then:*

- a bw-attacks b *if* $w(\langle a, x_1 \rangle) \cdot w(\langle x_1, x_2 \rangle) \cdot \ldots \cdot w(\langle x_n, b \rangle) < 0$.
- a bw-supports b *if* $w(\langle a, x_1 \rangle) \cdot w(\langle x_1, x_2 \rangle) \cdot \ldots \cdot w(\langle x_n, b \rangle) > 0$.

BWAF generalizes the notion of defense, based on a *multiplication rule*, so that: (i) it complies with the Dung-style property that even-length paths of attacks imply a defense (*the attack of an attack is a defense*); (ii) BAF notions of indirect attack and supported attack maintain their definitions.

As depicted in Figure 2.8, a BWAF can be represented as a digraph whose nodes are the arguments, attacks and supports are represented by solid and dashed edges respectively, and weights indicate the relative strength of relations. The work in [97] proposes a ranking semantics to evaluate argument acceptability in BWAFs. The approach relies on strength propagation of

indirect relations ending in each argument. Unfortunately, it only manages acyclic graphs, which is suitable for scenarios like discussion forums but may be inadequate for networks of agents in SWoT contexts.

The deductive argumentation framework proposed in Chapter 4 is a DL-based structured argumentation extension of the abstract BWAF framework.

Chapter 3

Cloud-Edge Artificial Intelligence

This chapter presents a novel cloud-edge AI microservice architecture based on the OC paradigm for IoT-oriented CPSs [85]. It enables the collection of data streams from nearby cyber-physical devices, preprocessing, AI model training and inference for ML classification and regression tasks. Particularly, AI microservices are encapsulated in containers and can be deployed without modification either to edge nodes or to cloud infrastructure. While cloud AI can manage larger datasets to maximize model accuracy and can offer additional analytics for end users, edge AI can deliver lower prediction latency and turnaround time, as well as inherent bandwidth savings and greater data privacy. The containerized AI service architecture allows for training and inference to be executed on the edge, cloud, or a combination of the two, taking advantage of available computational resources dynamically and opportunistically, with various trade-offs between computational/storage requirements and prediction accuracy.

One of the most significant roadblocks for novel edge AI proposals is the required high level of custom development and configuration. To overcome this issue, a direct mapping has been carried out between the proposed architecture components and Commercial-Off-The-Shelf (COTS) tools. This is facilitated by the microservice encapsulation of each architectural module with an accurate characterization of roles, responsibilities, and interactions,

thus increasing feasibility by lowering development costs and time to market. A fully working platform prototype has been implemented, using open-source software tools and commodity hardware, in order to demonstrate the viability of the proposal. A case study on cloud-edge AI in a smart manufacturing scenario and an experimental campaign validate the key value propositions of the approach from functional and performance standpoints.

3.1 State of the art

Edge Intelligence has been conceived as a paradigm that maximizes the performance of training AI models and inferencing, by fully exploiting the data and computational resources available throughout the hierarchy of end devices, edge nodes, and cloud datacenters [146]. EI applications can work in a cloud-edge device coordination, according to the six-level classification of architectures discussed in Section 2.1.1. Moving from "Cloud Intelligence" to "All On-Device", data privacy increases, while data offloading volume and path length decrease, as well as transmission latency and bandwidth cost. As a consequence, cloud and edge resources should be exploited opportunistically when designing complex pervasive cyber-physical systems.

The bulk of existing cloud AI platforms fall under the "In-Edge co-inference" level of the aforementioned classification, where model training takes place exclusively in the cloud and inferencing (prediction) is executed at the edge. A notable instance of this category is the *GEM-Analytics* platform for energy management: [130] it uses the cloud infrastructure to train and evaluate models, then periodically sends them to edge nodes, where they are used for day-to-day operations in power plants.

Osmotic Computing is one of the most actively studied approaches to overcome the difficulty to coordinate AI tasks across edge and cloud tiers. The main issues and challenges in designing and implementing AI-based applications in an OC environment are outlined in [91]. The OC paradigm has been employed in several works in a variety of contexts. The trust management framework for Pervasive Online Social Networks (POSNs) in [119] exploits OC for an efficient computational offloading among the many users of a POSN. An OC architecture is also proposed in [95] for a smart class-

room where deep learning models are exploited to control IoT devices and to recognize entities and chalkboard handwriting. However, since microservices are not containerized and a dynamic orchestration is missing, adopted OC features are quite basic. The *Apollon* OC platform, presented in [84] for pollution monitoring, allows for opportunistic filtering and integration of data from various mobile and IoT devices in urban environments. Similarly, by orchestrating microservices built on the *R* open-source statistical software, the *RAPTOR* [60] osmotic platform enables the development, deployment, and integration of customizable data analysis applications.

Recent works extend the core OC properties with more advanced capabilities. The framework *Osmosis* [134] focuses on microservices deployment across cloud, edge and IoT environments. Design principles of an osmotic smart orchestrator are investigated, capable of migrating *MicroELements* (MELs) composed of microservices along with *microdata* they work on. The aforementioned architecture is leveraged in [35] to accomplish a distributed healthcare system, and a Body Area Network (BAN) case study highlights key benefits of the approach. In [100] further orchestration mechanisms are proposed to implement a Message-Oriented Middleware (MOM) for IoT environments, based on OC principles. The framework *En-OsCo* [68] focuses on managing resources in an energy-aware manner. It uses a hyper-heuristic for the best service dispatch on incoming workloads and an extended Kalman filter to monitor edge datacenters. The Mobile Augmented Reality Network (MARN) architecture in [118] exploits OC for migrating and effectively scheduling various services across multiple servers. As they are crucial to support distributed mobile augmented/virtual reality applications, key requirements of low latency, robustness, and tolerance are tracked.

Table 3.1 summarizes relevant features of the above frameworks. In comparison, the proposed approach is the only one using both osmotic orchestration of containerized microservices and Cloud-Edge Intelligence, enabling to exploit predictive ML models trained and executed on edge and on cloud.

Table 3.1: Comparative table of frameworks (✓: supported, ✗: not supported)

Reference	**Containerized Services**	**Osmotic Orchestration**	**AI**	**Model Training**
Pacheco *et al.* [95]	✗	✗	✓	Pre-trained
Apollon [84]	✓	✓	✓	On cloud
Grzelak *et al.* [60]	✓	✓	✓	✗
Carnevale *et al.* [35]	✓	✓	✗	✗
En-OsCo [68]	✓	✓	✗	✗
Osmosis [134]	✓	✓	✗	✗
Sharma *et al.* [118]	✗	✓	✗	✗
Tovazzi *et al.* [130]	✓	✗	✓	On cloud
This framework	✓	✓	✓	On cloud and edge

3.2 Osmotic Cloud-Edge architecture

The architecture shown in Figure 3.1 serves as the foundation for the proposed framework. Microservices –denoted by little green cubes– are packaged in containers and opportunistically deployed to devices. A container only comprises specific components of the operating system (OS), middleware, and application-level software needed to run a certain (micro)service. Compared to employing a hypervisor, containers use OS-level virtualization to drastically reduce distribution overhead and increase instance density per device. Therefore, the new container-based techniques enable the deployment of lightweight services on resource-constrained programmable edge devices like gateways, network switches, and routers. They also improve the dynamic administration of microservices within cloud datacenters.

The orchestration and deployment of various containers on the available devices in the reference architecture adhere to the Osmotic Computing principles. Strategies for service orchestration consider the ever-changing requirements of both the infrastructure (such as load balancing, dependability, and availability) and applications (such as sensing and actuation capabilities, context awareness, topological closeness, and Quality of Service –QoS– characteristics). As a result, it is crucial to manage the migration of microservices between the cloud and the edge in both directions. Due to the

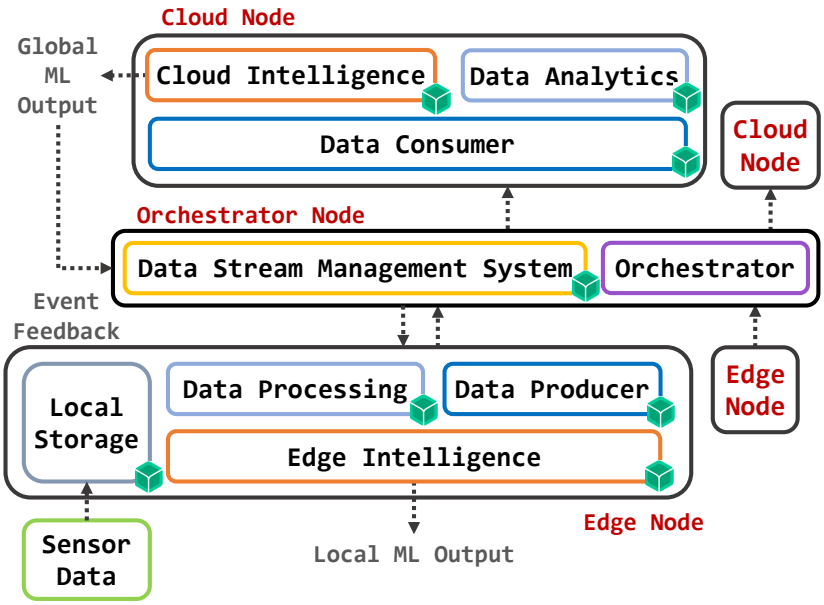


Figure 3.1: Osmotic Cloud-Edge architecture

significant degree of physical resource heterogeneity, the provisioning of containers must adapt the virtual environment to fit the destination hardware. Moreover, dynamic and effective management of virtual network resources is required for the transfer of services in the cloud-edge system in order to prevent application failures or QoS degradation. To address these problems, the management of data and applications (*data plane*) is separated from the control of network and security services (*control plane*). This strategy is supported by the OC paradigm, which offers an adaptable infrastructure for autonomous and secure provisioning of microservices.

The proposed architecture, as depicted in Figure 3.1, spans two main infrastructure layers: cloud and edge. In the cloud, datacenters host various service kinds that are built in accordance with high-level application needs. The edge-level computing environment, located between neighborhood IoT devices and the Internet, consists of gateway nodes and data acquisition points that may process data generated by end devices. The latter collect raw data with frequencies depending on a variety of factors, including:

- rate of change of the observed phenomenon;

- environment and context constraints;
- capacity of the device itself to gather or store data;
- operational system requirements to be met.

Due to computing resource limitations, only lightweight data preprocessing is possible on edge components, acting on the gathered raw data to prepare them for mining workflows:

- encrypting outgoing and decrypting incoming data streams for security;
- transcoding data streams between different formats;
- integrating various data streams from groups of devices;
- preprocessing and filtering streams to remove spurious data, noise and artifacts;
- summarizing raw data to reduce volumes with minimal information loss.

In traditional solutions, the most sophisticated and computationally challenging operations are allocated to the cloud infrastructure, including:

- advanced input data stream preprocessing, such as transform functions to representations in the frequency domain;
- feature extraction and selection for data dimensionality reduction;
- model training from features;
- prediction using the trained model.

Conversely, in the architecture herein described, edge nodes are capable to run both prediction-only tasks using pre-trained models and the complete feature extraction–training–prediction workflow. Due to the various requirements and capabilities of cloud and edge systems, it is appropriate to design a heterogeneous architecture where different types of resources are distributed on the two layers. In this perspective, components at the edge level usually

have considerably lower processing and memory capacity, therefore model training tasks can leverage larger datasets in cloud w.r.t. edge deployments. As a result, model training and prediction should be deployed to the edge when prediction accuracy is less important than other criteria, such as reducing response latency or maintaining data locality owing to privacy concerns; in those situations, suboptimal accuracy is an acceptable trade-off.

Due to the direct relationship between container performance and the capabilities of the physical host, it is crucial to define the way composite microservices must be dynamically adapted to deployment sites, taking location and distribution context into account. As depicted in Figure 3.1, the following logical components are used to accomplish a dynamic service orchestration based not only on restrictions specified by the particular application and by the infrastructure provider, but also on a real-time feedback loop to detect changes in infrastructure performance and QoS metrics:

- one or more *edge nodes* programmed to acquire raw data and to process them locally using machine learning algorithms for classification or regression tasks;
- one or more *cloud nodes* able to receive aggregate data from the edge and perform classification/regression tasks by operating on a larger and more feature-rich data set, while also being able to act as backup hosts for edge microservices in the case of unavailability of edge nodes *e.g.*, due to failures or energy depletion;
- a *Data Stream Management System* (DSMS) capable of conveying data coming between the edge of the network and the cloud modules, while also providing support for data storage operations;
- an *Orchestrator*, following the OC paradigm to manage the different containers implementing the required functional blocks as microservices.

Individual components are detailed in the next subsection, while technological choices for their reference implementation and integration are explained in the subsequent one.

3.2.1 Microservices

The proposed architecture relies on a number of lightweight services that are loosely connected, enable granular scalability, and support flexible composition patterns to satisfy the requirements and constraints of applications. The following services have been defined.

Local Storage: stores locally and temporarily the data gathered from IoT field devices. For the sake of latency and bandwidth optimization, centralized and shared data storage should be avoided. In fact, each data processing service requires the data to be located as close as possible. This service provides simple access mechanisms like RESTful (REpresentational State Transfer) APIs (Application Programming Interfaces) and event-driven interaction.

Data Processing: performs preprocessing for subsequent ML model training tasks. For complex and high-volume CPSs, the overall task is distributed among edge nodes to spread the computational load and exploit data locality. In fact, data from the local storage service are accessed directly, thereby reducing bandwidth consumption and latency, while also mitigating common issues of edge devices, such as power and connectivity outages.

Data Stream Management System (DSMS): acts as Message Broker (MB) for the platform, forwarding data and event streams from edge to cloud and vice versa. It adopts the *publish/subscribe* pattern, enabling efficient event-driven asynchronous communication. This paradigm is particularly well suited for microservice architectures [100]. Each Edge node can send messages to unique *topic*s marking different information types. Each topic has zero to many consumers subscribing to it, and refers either to raw or preprocessed data streams, or to events and control messages. The DSMS allows also the discovery of available topics, published by data producers at the edge. Furthermore, it acts as an event stream processor, by combining and possibly converting input data from multiple selected topics to produce an output flow which is subsequently processed by the Cloud Intelligence modules to train a global model. It is crucial for the DSMS to be interoperable with the most widespread IoT communication protocols, such as *MQTT* (Message Queuing Telemetry Transport) [14] or *CoAP* (Constrained Appli-

cation Protocol) [121], as many kinds of commercial IoT devices cannot be upgraded to support new protocols due to proprietary firmwares as well as computational resource and deployment capability limitations.

Data Producer: sends data from an edge node to other edge or cloud nodes. The message broker supports the connection.

Data Consumer: receives data sent by a data producer. It is typically deployed on cloud nodes to get preprocessed information from edge nodes, in order to be mined.

Edge Intelligence: executes algorithms on Edge devices for ML problems like classification and regression. This microservice is also able to provide model training and validation, based on the data provided by the local storage. The following benefits ensue: (i) privacy and security, as the transfer of sensitive data across the Internet can be avoided; (ii) low latency, as the local model can be trained without uploading data to the cloud and downloading models or prediction outcomes; (iii) scalability, as distributed learning is able to manage high volumes of data produced in pervasive IoT-based CPSs.

Cloud Intelligence: the cloud counterpart of the Edge intelligence service. It runs ML algorithms on data streams gathered through the edge layer. For example, the cloud node can train a classification or regression model on streamed sensor data, collected from multiple data producer instances. This approach enables a feedback control loop to update the model and improve its quality progressively; a less accurate model can be trained and used on edge devices, while a more accurate one is trained in the cloud by collecting larger amounts of data and is then transferred to the edge. This loop can be repeated periodically when new data batches are collected.

Data Analytics: carries out further business intelligence analytics on data gathered at the cloud layer. In particular, it provides functionalities and tools to support a presentation layer, *e.g.*, a dashboard where aggregated statistics as well as predictions and performance of trained models can be reported.

Orchestrator: manages the aforementioned microservices by means of a container-based approach. In particular, it schedules the migration of services from edge to cloud and vice versa, based on real-time resource conditions and

availability. For example, the orchestrator can reassign containers in case of network infrastructure changes, high service demand or edge node failures.

3.2.2 Technologies

One of the main objectives of the presented platform architecture is to facilitate the realization and integration of autonomous microservices by utilizing commercial-off-the-shelf software components to implement both system functionalities and fundamental application modules, allowing for reduced platform development time and effort.

Table 3.2: Reference COTS tools

Service/ Module	**Technology**	**Version**	**License**	**Release Date**
Container technology	balenaOS	2.54.2	Apache 2.0	Aug 12, 2020
Orchestrator	openBalena	3.1.1	GNU Affero GPL 3.0	Nov 10, 2020
Data Stream Management System	Apache Kafka	2.5.0 (with Scala 2.12)	Apache 2.0	Apr 15, 2020
Data Producer	Kafka Producer API	2.0.1-python	Apache 2.0	Feb 19, 2020
Data Consumer	Kafka Consumer API	2.0.1-python	Apache 2.0	Feb 19, 2020
Local Storage	Redis	6.0.9	3-Clause BSD	Oct 26, 2020
Data Processing	Python scripts	3.9.0	PSF & Zero-Clause BSD	Oct 5, 2020
Edge/Cloud Intelligence	TensorFlow Keras API	2.3.1 2.4.3	Apache 2.0 MIT	Sep 24, 2020 Jun 25, 2020
Data Analytics & Visualization	Streamlit	0.72.0	Apache 2.0	Dec 2, 2020

Table 3.2 illustrates the mapping of each architecture component (Figure 3.1) with the corresponding selected technology. To choose tools appropriate for each microservice, a comprehensive and in-depth market investigation has been conducted, analyzing functionality, technological characteristics, pricing, licensing, and hardware and software requirements. COTS components with the following features have been preferred:

- open source software license;
- active developer community;
- proven track record of reliability, security and performance;
- full compatibility with container technologies;
- interoperability with widespread IoT platforms and protocols;
- support for multiple hardware architectures;
- compliance with cutting-edge functional and architectural paradigms of software engineering.

The following technologies have been selected to implement and integrate the proposed platform:

balenaOS:[1] a lightweight operating system based on the *Yocto* project[2] for Linux distribution customization. balenaOS is tailored to run application containers on single-board computers and embedded devices. The OS provides robust networking functionalities as well as virtualization and provisioning support. For container management balenaOS includes *balenaEngine*, a *Docker*[3]-compatible daemon optimized for application service images, containers and volumes deployed on resource-constrained devices. With respect to other existing container technologies, this tool overcomes common virtualization problems related to embedded scenarios such as resource overhead and lack of hardware support, as balenaOS is available for several device types and different CPU architectures.

openBalena:[4] a balenaOS-based provisioning and orchestration platform to deploy and manage containers on fleets of devices. It is exploited to configure application containers, push updates, share network parameters and distribute container images on each device according to multiple strategies.

[1] `https://www.balena.io/os`
[2] `https://www.yoctoproject.org`
[3] `https://www.docker.com`
[4] `https://www.balena.io/open`

Apache Kafka:[5] a distributed event streaming platform for communication among several devices and applications, characterized by horizontal scalability, high throughput, low latency and interoperability with existing IoT communication protocols through an ecosystem of plug-ins and connectors. Kafka has been adopted as DSMS for sending/receiving streams of event data collected by the container applications. Messages can also contain event feedback forwarded to edge nodes and the outputs of the ML algorithms exchanged between cloud and edge microservices.

Kafka Producer/Consumer API:[6] each microservice can produce (*i.e.*, send) or consume (*i.e.*, receive) data through the Kafka API. The Producer API allows containers to send data to other services subscribed to the same topics, whereas the Consumer API can be exploited to retrieve information marked with specific topics in the Kafka platform. Both APIs are available for several programming languages; Python[7] has been chosen as it facilitated integration with other Python-based platform components, like the ML APIs and the custom scripts developed for the Data Processing microservice.

Redis:[8] in-memory data store used to collect information coming from sensors and field devices according to a key-value data model. Several features make it appropriate for Edge Computing scenarios: (i) low CPU and memory requirements; (ii) lightweight data structures particularly appropriate for time-series data; (iii) simple but versatile data model, useful to store information produced by heterogeneous devices; (iv) append-only storage options optimized for flash memories, usually adopted in IoT devices.

TensorFlow:[9] an open source machine learning library, exploited to process the collected data on both edge devices and cloud nodes. The *Keras*[10] high-level API has been used to define and train classification and regression models based on deep neural networks, as well as to make predictions on data.

[5] `https://kafka.apache.org`
[6] `https://kafka.apache.org/documentation/#api`
[7] `https://www.python.org/`
[8] `https://redis.io`
[9] `https://www.tensorflow.org`
[10] `https://keras.io`

Streamlit:[11] Python-based library used to create interactive Web applications able to: (i) plot sensor data, also highlighting basic statistics and patterns; (ii) support exploratory data analysis; (iii) visualize performance results of ML predictive models.

3.3 Case study

Smart manufacturing is a challenging scenario for cyber-physical systems. Several *Industry 4.0* [94] initiatives are accelerating the adoption of IoT and AI technologies, changing manufacturing with significant organizational, societal, and economic impacts. Extensive networks of smaller, individually configurable sensing and actuation components are replacing collections of large, monolithic machines and robots in plants. Multiple continuous data streams are feeding numerous distributed decision points, either autonomous or under human supervision.

The reference setting considered for the case study involves impurity prediction on iron concentrate in the mining industry. *An iron extraction plant operates in an industrial area. Process variables and surrounding air flow must be continuously monitored, carrying out an autonomous intelligent manufacturing task to maximize mineral quality.* Mining activities face a constant decrease in ore concentration. Various processing methods are employed nowadays to enhance the recovery of ore from raw materials and are key components of contemporary separation processes utilized in the mining sector. *Flotation* is one of the most widely used techniques, allowing the separation of gangue from ore. However, since the impurity (silica) in iron ore is commonly measured every hour, being able to predict the amount of impurity could constantly support the activity of engineers and technicians, by providing useful information in advance to promptly improve the extraction process.

A prototype of a monitoring platform has been created adopting the proposed architecture. Sensors and actuators embedded in the extraction plant have been simulated by means of the dataset in [87]. Inspection features

[11] `https://www.streamlit.io`

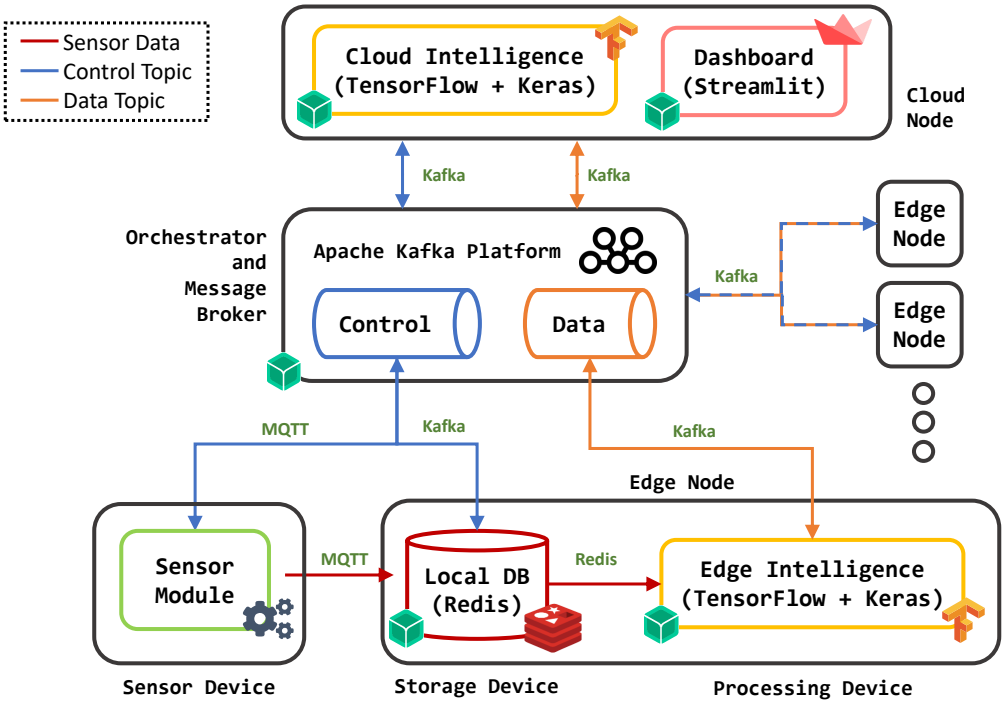


Figure 3.2: Components of the proposed prototype

include starch and amina (reagents) flow, ore pulp flow, ore pulp pH, and ore pulp density, which are the most important variables for the final mineral quality. Further data include level and air flow inside the flotation columns, *i.e.*, cylinders where mineral slurry and air flows are introduced from above and from below, respectively, in order to induce mixing. The proposed architecture allows for opportunistically combining this data and predicting the amount of silica that will be extracted using Cloud-Edge Intelligence algorithms.

3.3.1 Prototype implementation

In accordance with the architecture discussed in Section 3.2, a prototype testbed has been developed to demonstrate the feasibility of the proposal and assess its performance and capabilities. Figure 3.2 shows the main elements of the industrial IoT-based CPS environment: a cloud node, an orchestrator and Message Broker (MB in the following), and two edge nodes, representing

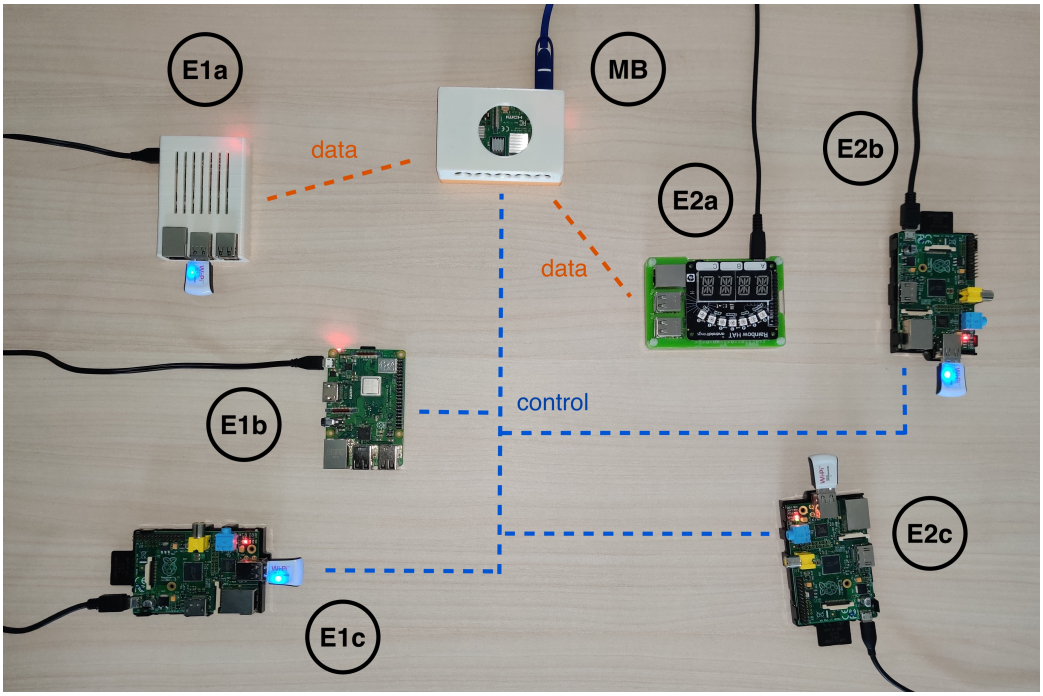


Figure 3.3: Message Broker/Orchestrator and Edge devices in the platform prototype

two distinct plant sectors. Edge devices and the MB are connected through an IEEE 802.11 wireless local network, and the MB is the only module communicating with the remote cloud node through an Internet connection.

Various single-board computers have been used on the edge side, as shown in Figure 3.3. Considering Figure 3.2, each microservice has been deployed on a distinct board: this demonstrates how logically related microservices can be distributed over several hardware devices in the proposed framework rather than being contained in a single node. The system scales horizontally according to available devices within the target environment. In contrast to boards with higher-performance CPUs, which can be employed for more computationally demanding activities, devices with lower computing capacity may be used as Storage modules, collecting data from physical sensors. High modularity and scalability are the main benefits of the adopted OC approach.

A Raspberry Pi 4 Model B[12] (RPi4) has been used as the MB, running

[12] `https://www.raspberrypi.org/products/raspberry-pi-4-model-b`

an instance of Apache Kafka, a MQTT-to-Kafka connector, and the Orchestration service in different balenaOS-based containers. It is equipped with a quad-core 1.5 GHz ARM64 CPU, 4 GB of RAM, and 32 GB of Secure Digital (SD) storage memory. As per the publish/subscribe pattern, each message published by any node to a topic is received by all subscribers for the topic. The following topics have been defined: `control`, used to transmit messages regarding the (dis)connection of Sensor nodes, or data availability in a Storage module; `data`, used to share messages related to data processing and results of the inference algorithms.

Each edge node as depicted in Figure 3.3 is composed of two devices running different containerized modules on balenaOS. The first edge node (E_1) includes a Raspberry Pi 3 Model B+[13] (RPi3+) to perform Edge Intelligence tasks (E_{1a}) and a local Storage module running on a Raspberry Pi 1 Model B (RPi) (E_{1b}). The RPi3+ is equipped with a quad core 1.4 GHz ARM64 CPU, 1 GB RAM, and 32 GB storage memory, whereas the RPi has a single-core ARM11 CPU at 700 MHz, 512 MB RAM, and 8 GB SD storage memory. Similarly, the second edge node (E_2) has been configured using a Raspberry Pi 3 Model B[14] (RPi3) equipped with a slightly slower quad core 1.2 GHz ARM64 CPU, 1 GB RAM and 32 GB SD storage memory (E_{2a}), which runs the Edge Intelligence service, and a RPi acting as a second Storage device (E_{2b}). Two additional RPi devices (E_{1c} and E_{2c}) with the same specifications replicate actual sensing devices in a sensor network, transmitting raw data to the Storage modules via dedicated MQTT topics.

The cloud node containers have been deployed to a Microsoft Azure D32as v5 virtual machine, configured with an Intel Xeon CPU E5-2673, 32 GB of RAM and 128 GB of storage. The Internet connection between the local network (including the Message Broker) and the Cloud is an asymmetric Fiber To The Cabinet (FTTC) small office link with downstream and upstream nominal bandwidth of 100 Mbps and 30 Mbps, respectively.

Different message formats have been used for communication. Simulated Sensor devices communicate with the MB to discover available Storage mod-

[13] `https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus`
[14] `https://www.raspberrypi.org/products/raspberry-pi-3-model-b`

ules, then they start transmitting data serialized in the *Apache Arrow*[15] format, a language-independent standard proposed for general-purpose serialization and data transfer. Since it is based on a column-oriented layout, the Arrow format is particularly suitable for tasks requiring fast data processing and information sharing between Storage devices. On the other hand, all messages transmitted through the MB via the `control` and `data` topics are serialized in *JSON* (JavaScript Object Notation)[16] format. In particular, `control` messages have the following attributes, summarized in Table 3.3:

- *id*: unique message identifier;
- *type*: indicates the kind of control message. Acceptable values are:
  - *storage_connected* (SC): a new Storage module is available on the network;
  - *storage_disconnected* (SD): a Storage module is currently unreachable or down;
  - *sensor_data* (SDT): a Sensor measurement is available on a storage module for running a prediction algorithm;
  - *dataset* (DS): a Sensor dataset, including several sensor measurements, is available on a Storage module for training or updating the ML models;
  - *query* (QR): used to list available Storage modules and related data;
  - *response* (RS): indicates a response to a query message.
- *host*: contains the reference module IP address;
- *data_key*: unique identifier used to retrieve data from a specific Redis datastore;
- *query_type*: used to retrieve information about a single measurement (*sensor_data*), a subset of data (*dataset*) or the whole collection (*storage*);

[15] `https://arrow.apache.org`
[16] `https://www.json.org`

- *query_id*: message id of the query originating current response;
- *storage_id*: unique identifier of the Storage module containing the data.

Table 3.3: Attributes of control message

Type	**Id**	**Host**	**Data Key**	**Query Type**	**Query ID**	**Storage ID**
SC	✓	✓				
SD	✓					
SDT	✓	✓	✓			
DS	✓	✓	✓			
QR	✓			✓		
RS	✓				✓	✓

Table 3.4 summarizes the attributes of `data` messages:

- *type*: indicates the kind of data notification:
  - *input*: contains data samples for which an inference task is requested;
  - *output*: contains results of a prediction task;
  - *model*: returns information about the performance of the trained ML models.
- *id*: identifies the processed sensor data (in case of input and output messages) or the Cloud/Edge Intelligence module providing the prediction model;
- *data*: an array of raw information;
- *module_id*: identifies the Intelligence node running the predictive algorithm;
- *result*: output of the prediction task;
- *time*: prediction time in milliseconds;
- *r2*: coefficient of determination (R^2), used as a performance metric for a regression model;

- *mse*: mean squared error, *i.e.*, the average squared difference between predicted and real values;
- *download_time*, *training_time* and *evaluation_time*: time spent by the Intelligence node to retrieve the whole dataset, train the model and evaluate performance, respectively.

Table 3.4: Attributes of data messages

Type	input	output	model
Id	✓	✓	✓
Data	✓		
Module ID		✓	
Result		✓	
Time		✓	
R^2			✓
MSE			✓
Download Time			✓
Training Time			✓
Evaluation Time			✓

An ore purity monitoring and prediction process has been developed with a regression model through the above prototype. Basically, it entails the sequence of interactions reported in the UML (*Unified Modeling Language*) diagram of Figure 3.4 and described in what follows:

1. when a Storage module is available on the network, it sends a *storage_connect* control message to notify all Sensor modules subscribed to the `control` topic (blue messages in Figure 3.4). As an alternative, each Sensor module can explicitly perform a query to retrieve all the available Storage devices;
2. the Sensor module collects data during its observation period and sends them to Storage devices through a dedicated MQTT topic. The Apache Arrow data format is used for message serialization (red color in Figure 3.4);
3. a *dataset* notification is sent to advertise the availability of new data. Datasets can be used to train or update prediction algorithms on active

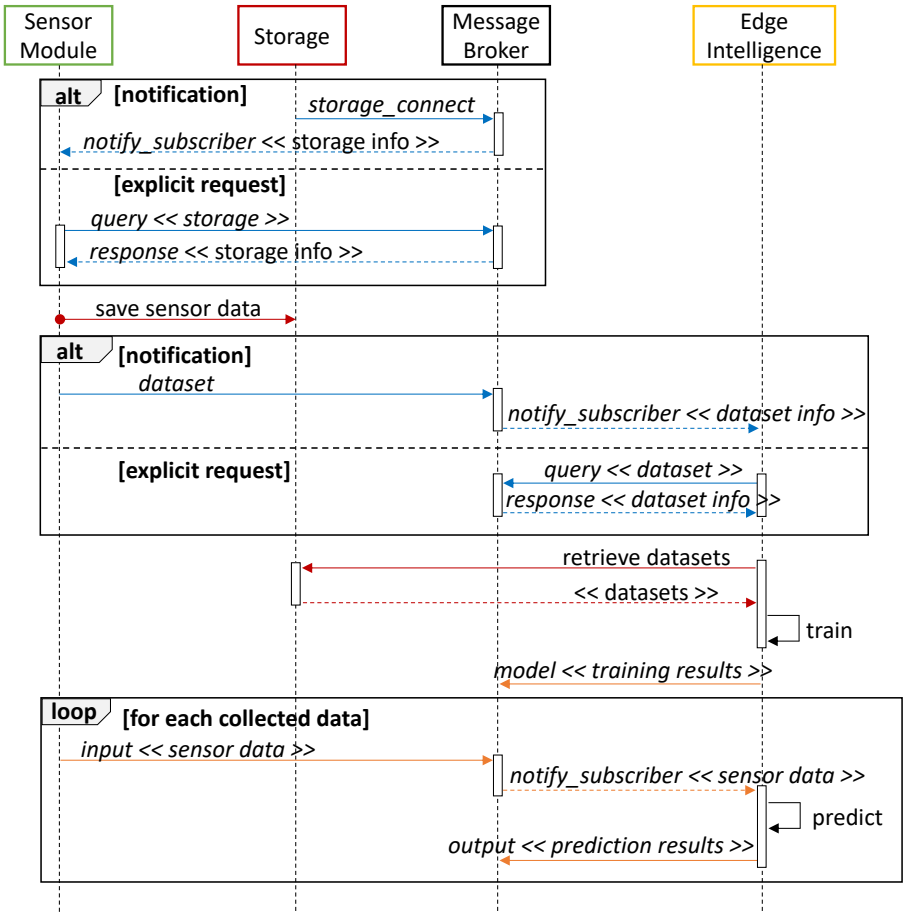


Figure 3.4: Sequence diagram for edge-side prediction

Cloud/Edge Intelligence modules, but also to plot information on a remote dashboard. Intelligence modules can autonomously query the MB to obtain information about available datasets;

4. data are retrieved from one or more Storage devices and used to train a regression model. Performance results are then exposed through a *model* message on the `data` topic (drawn in orange in Figure 3.4);

5. subsequently collected sensor data represent the *input* of the prediction model and are forwarded through the MB to the subscribed Intelligence nodes. Results of the regression process are finally returned through

an *output* notification.

Final prototype specification concerns the ML model trained in the Intelligence nodes for the case study: it is a *multi-layer perceptron* regressor [59] with 5 hidden layers and 200 neurons per layer, with a *Rectified Linear Unit* (ReLU) activation function. The network is trained for 10 epochs using the Keras implementation of the *Adam* [73] optimizer, with default parameters and mean squared error loss function. This model has been selected since it generally provides satisfactory prediction performance, while being sufficiently lightweight to run in a timely manner on resource-constrained devices.

3.4 Experimental evaluation

An experimental campaign has been carried out to assess the prototype performance. The whole set of data [87] adopted to simulate the intelligent manufacturing use case contains $N = 737453$ samples, collected in a 7-month time span, for a 160 MB total size. The deployed architecture is the one described in Section 3.3.1, where each logical node is composed of two devices: Edge Intelligence and Storage. In order to test the dependency of performance on dataset size and to simulate deployments on a larger scale than what was allowed by available physical devices, four scenarios have been configured, with data dimension set to $N, \frac{N}{2}, \frac{N}{4}, \frac{N}{8}$ respectively, and samples extracted randomly. In each configuration, a validation set has been obtained by holding out 1/7 of the dataset, and the remaining 6/7 have been used to train the predictive models, with 3/7 for the two simulated Sensor devices, E_{1c} and E_{2c}. With regards to all experimental results, each reported value is the average of five cold runs.

Data gathering

The first test has simulated data upload by Sensor devices to Storage. To reduce network load and memory usage, due to Redis being an in-memory store, data have been first compressed using the *zlib* format [45]. Basically, compression increases CPU usage on both Sensor and Edge devices, while

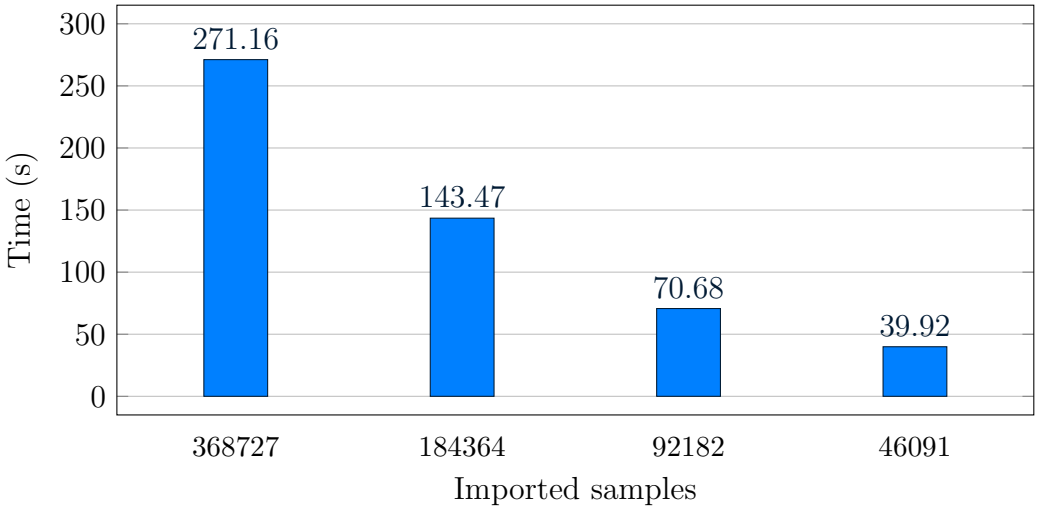


Figure 3.5: Time required for data import

a range of network bandwidth and RAM to CPU usage trade-offs can be achieved by tuning the compression parameters, or by replacing the format altogether, so as to meet scenario requirements. Data upload has been carried out according to steps 1-3 of the sequence in Section 3.3.1. Figure 3.5 reports elapsed times for each scenario. Results show linear dependence of import time on the number of samples.

Model training and validation

The second test has involved measuring training times and the related network load. As explained in steps 4-5 of the sequence in Section 3.3.1, each Intelligence module needs to fetch data from Storage before training. On training completion, it performs predictions on the validation set. Table 3.5 reports on network loads for this phase, with device labels referring to Figure 3.3. Predictably, the busiest modules from the network point of view are the Storage ones, which both receive information from Sensors and upload them to requesting nodes. Despite appearing inefficient, this method separates data generated by field devices from data consumed by Edge Intelligence modules, which is convenient as their variability and/or velocity may differ greatly.

Table 3.5: Training: Network activity

Node	Device	Label	Download (kB)	Upload (kB)
Broker	broker	MB	4768	7603
Edge 1	intelligence	E_{1a}	33386	993
	storage	E_{1b}	35951	36043
	sensor	E_{1c}	2379	35259
Edge 2	intelligence	E_{2a}	33685	1142
	storage	E_{2b}	35522	35858
	sensor	E_{2c}	2193	35088

In order to compare the performance of the proposal against a centralized cloud solution, the same Intelligence container used in the edge nodes has been deployed on the cloud via the OC Orchestrator on the MB node. The cloud node first downloads the full dataset by querying all the edge Storage devices, then proceeds to train a predictive model. Turnaround times and model validation results are reported in Table 3.6. Considering that edge nodes use half of the samples to train their models, it can be noticed how Storage data retrieval takes a comparable amount of time for both cloud and edge nodes, while training time is significantly shorter on the cloud (almost an order of magnitude), as expected due to its more powerful hardware. While this may be mitigated by deploying Edge Intelligence services on more capable devices, it is not a crucial issue, as training on the whole dataset only happens once, or at worst periodically, depending on the use case. Actually, in real applications it is advisable to train models periodically and incrementally, using small-size datasets. As explained in Section 3.2, the prototype also allows for cloud-edge cooperation in model training: while edge components can train models "on-the-fly" on smaller datasets, retaining their independence from the Cloud albeit with suboptimal accuracy, a cloud node may aggregate more data coming from multiple local networks in order to train better models, finally feeding them back to the edge. It is also important to point out that the prototype represents a rather optimistic setting for the Cloud node, where it is mostly idle w.r.t. hardware and network resources. In a realistic industrial scenario, where the premises Internet connection towards the cloud is shared by a large number of devices,

the uplink may be temporarily unavailable or may be saturated by sensor data and inference requests, making it a potential bottleneck. Similarly, the full processing resources of the cloud node(s) will be shared across highly heterogeneous workloads, and company budget pressures will induce IT officers to seek a relatively high utilization baseline [25]. In such settings, Edge Intelligence capabilities can improve both the availability and the timeliness of predictions, by scaling the workload across multiple relatively capable boards located near data generators.

Table 3.6: Training time and validation results

Node	**R^2**	**MSE**	**Download Time (s)**	**Training Time (s)**	**Validation Time (s)**
Cloud	0.983	0.0222	50.788	437.986	3.081
Edge 1	0.972	0.0348	24.603	2086.653	25.415
Edge 2	0.971	0.0337	33.085	2574.005	28.976

Dataset size dependency

Table 3.7 reports on training times and validation metrics (R^2 and MSE) for different dataset sizes on the Edge Intelligence node E_{1a}. Results indicate there is an acceptable trade-off between model accuracy and training time in Edge Intelligence applications, in agreement with existing evidence suggesting how fractional datasets do not induce a large degradation in prediction accuracy if their distribution is representative of the whole dataset [4]. This outcome supports the aforementioned claims about Cloud-Edge Intelligence cooperation. Additionally, it is important to note that the prototype has been set up with separate devices for data storage and model training for experimentation purposes. In real scenarios, the OC Orchestrator may eliminate download latency by deploying Edge Intelligence and Storage microservices on the same component, provided it has enough resources; this would both eliminate download time and reduce the overall network load.

Table 3.7: Training on Edge node: reduced datasets

Samples	**R^2**	**MSE**	**Download Time (s)**	**Training Time (s)**	**Validation Time (s)**
737454	0.983	0.021	42.451	5155.740	32.072
368727	0.972	0.035	24.603	2086.653	24.608
184363	0.959	0.055	10.688	1066.539	10.688
92182	0.931	0.088	5.087	564.569	5.087
46090	0.894	0.135	2.815	263.953	2.815

Table 3.8: Prediction: Network activity

Node	**Device**	**Label**	**Download (kB)**	**Upload (kB)**
Broker	broker	MB	42	78
Edge 1	intelligence	E_{1a}	26	26
	sensor	E_{1c}	36	34
Edge 2	intelligence	E_{2a}	26	26

Table 3.9: Prediction: time and latency

Node	**Inference Time (ms)**	**Communication Latency (ms)**				**Turnaround Time (ms)**
		S to MB	**MB to I**	**I to MB**	**MB to S**	
Cloud	31.377	91.510	19.752	42.549	44.970	230.158
Edge 1	230.598	87.259	4.362	19.461	37.773	379.453
Edge 2	301.887	84.812	7.590	22.844	30.555	447.688

Prediction performance

Further experiments have been carried out to evaluate both computational and network performance in prediction tasks. Sensor device E_{1c} has been configured to send 10 *input* messages on the *data* topic. Subscribed Intelligence nodes compute predictions and return *output* response messages. Table 3.8 shows the data exchange for this test is minimal, as expected. Sending individual data samples through the MB incurs in some network bandwidth overhead, though this can be mitigated by publishing multiple samples together, if possible. The last experiment has assessed prediction time and latencies both at the edge and on the cloud. Outcomes are reported in Table 3.9 and are described in what follows:

- *inference time*: the time elapsed in predicting the regression value for a sample locally, as measured by the Intelligence module;
- *communication latency*: the time required for sending and receiving messages between the different components of the architecture in the prediction phase. As Table 3.9 shows, it is made of four components: (i) from Sensor to Message Broker (S to MB), (ii) from Message Broker to Intelligence (MB to I), (iii) from Intelligence to Message Broker (I to MB), and from Message Broker to Sensor (MB to S). In the prototype the last two components simply concern the prediction values, but in general scenarios they could concern set points for appropriate actuators in a control feedback loop, computed on the basis of the ML predictions;
- *turnaround time*: is the overall time between input sample upload and prediction, evaluated on the Sensor node uploading the samples.

While prediction time is significantly lower on the cloud, the edge nodes have notably lower communication latencies. Although the overall turnaround time is lower in the cloud case, the actual availability of a stable connection is far from certain in a real industrial plant. In fact, whenever bandwidth availability decreases due to other computing and/or control activities deriving from large numbers of local devices interacting with the cloud –a condition which is not replicated in the experimental setup–, transferring training and

prediction to the edge may improve the overall system performance by alleviating some of the outbound network pressure.

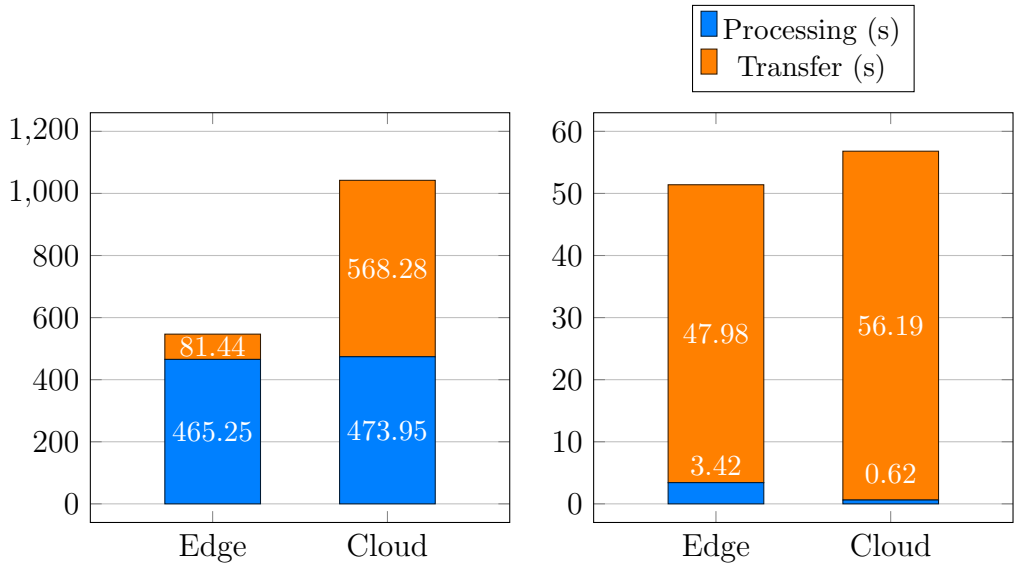
Osmotic microservice allocation

In order to assess the capability of the proposed architecture to dynamically adapt to variable workloads and node availability by migrating microservices between the edge and the cloud, two alternative scenarios to the baseline behavior described in Section 3.3.1 have been tested:

1. A Sensor module looks for an available Storage service, but the Orchestrator is unable to meet the request due to the unavailability of Storage microservice instances and to the lack of a suitable device to host them. The Orchestrator therefore pushes the Storage container to the cloud, which –once ready– announces itself through a *storage_connect* message. The Orchestrator is now able to notify the Sensor module, which can then upload its data. At some point in time, a new edge device with the required capabilities to act as a Storage host connects to the network. For load balancing purpose, the Orchestrator hangs the Storage microservice to the new device, which can take over the role of Sensor data collector.

2. In case of a shortage of Edge Intelligence nodes (*e.g.*, due to device failure), the Orchestrator pushes the Intelligence microservice to the cloud, where it can resume its learning and inference tasks, albeit with higher network latency. Eventually, a new device connects at the edge, and the Orchestrator assesses it as being able to host an instance of the Edge Intelligence service. The microservice is therefore linked to the new device, thus offloading the cloud and restoring normal operation.

The main difference from the baseline essentially consists of the Orchestrator pushing containers to cloud and edge nodes, while the rest of the system keeps working as usual. Microservice deployment times and bandwidth usage are reported in Table 3.10: the *load time* column reports on the time taken to address the container from the Orchestrator to the target node and load it in balenaEngine. Conversely, *startup time* refers to the time it



(a) Upload (b) Download

Figure 3.6: Dataset transfer times

takes to bootstrap the container after it has been loaded. Both load and startup times are higher for edge nodes, as expected due to being significantly less capable, but still within one order of magnitude of their cloud counterparts. Times for the Intelligence container are much closer, which is likely due to the allotted edge device being a RPi3+, versus a RPi for the Storage container; this hints at the dominance of container loading time in balenaEngine over network transfer time for slower devices. Bandwidth usage has been measured on the Orchestrator, and it is generally lower for the edge node: this is due to the size of container images being different between the target devices, as reported in the *image size* column.

Table 3.10: Microservice deployment performance

Node	**Service**	**Load Time (s)**	**Startup Time (s)**	**Broker Bandwidth (MB)**		**Image Size (MB)**
				Download	**Upload**	
Cloud	Storage	71.163	2.983	4.144	255.726	239.974
	Intelligence	224.451	2.768	12.648	765.300	718.205
Edge	Storage	314.577	23.424	1.713	149.924	137.782
	Intelligence	244.422	3.108	3.424	718.291	694.802

Finally, Figure 3.6 shows dataset upload and download times when the

Storage microservice is deployed to the edge and cloud nodes, representing Sensor data compression and upload in the first scenario, and dataset download and decompression by Edge Intelligence nodes in the second scenario, respectively. Processing time is mostly noticeable in the upload phase, as data is compressed by an RPi device before uploading them to the Redis data store, while it is negligible when an RPi3 device decompresses the downloaded dataset. Transfer time is significantly higher for the cloud node, which is due to the network connection being asymmetrical, with much higher downlink than uplink bandwidth. In any case, transfer time is consistently higher when the Storage microservice is deployed to the cloud, as expected, though again in the same order of magnitude. This confirms that the elastic –osmotic– allocation of the Storage microservice between the edge and the cloud is a viable solution which can be carried out without significant impact on the overall framework performance.

Chapter 4

Deductive Argumentation Framework

This chapter introduces a novel approach for improving argumentation frameworks adopting Description Logics-based Knowledge Representation and Reasoning to represent arguments, evaluate their relations and rank their acceptability [56]. The approach is general-purpose, but it is particularly tailored to SWoT-enabled EC contexts, like the ones supported by the framework discussed in the previous chapter.

Information generated and shared by an agent is annotated according to reference ontologies in standard Semantic Web languages, grounded on Description Logics [13]. Specifically, the framework refers to a Web Ontology Language (OWL) 2 [96] subset corresponding to the *Attributive Language with unqualified Number restrictions* ($\mathcal{ALN}$) DL, which allows polynomial-time standard and non-standard inference procedures for acyclic ontologies. The approach can be integrated with any classical Abstract Argumentation framework, where each semantic annotation takes the role of an argument. However, in order to allow more nuanced analysis, the proposed approach adopts a *Bipolar Weighted Argumentation Framework*, as discussed in Section 2.3.2, including both attack (*a.k.a.* defeat) and support relations in the interpretation introduced by [36], with an assigned weight to represent connection type and strength. The type and weight of relations are computed by means of non-standard, non-monotonic inference services including *Concept*

Contraction, *Concept Abduction* and *Concept Bonus*, [104] recalled in Section 2.2.2. This enables a more meaningful appraisal of individual relations and interpretability of results, by virtue of logic-based explanation capabilities of the exploited deductions. The framework includes a novel *propagation-based ranking semantics* to compute an acceptability score for each argument. The proposed method adds a *fading* mechanism to path strength propagation, in order to cope with agents' computational resource limitations. Unlike most existing BWAF proposals, it also copes with cyclic argumentative graphs, by using an iterative algorithm with halt conditions.

In order to clarify the proposal and highlight its peculiarities, a case study concerning tactical decision-making in the Real-Time Strategy (RTS) game engine *StarCraft II* has been conducted. The choice is due to the similarity of RTS game environments with real-world pervasive contexts from unpredictability and complexity standpoints. While general purpose, a significant motivation for the framework is to be feasible even in pervasive computing contexts. Hence, an approach for argumentation graph simplification via edge pruning is proposed: its effects have been investigated w.r.t. both reduction in computational overhead and stability of the outcomes of the ranking semantics, in order to evaluate whether it can provide an acceptable trade-off between approximation of results and resource consumption, as is often the case in Edge Intelligence applications on resource-constrained devices [44].

The KRR-based argumentation framework can leverage the Cloud-Edge AI architecture proposed in Chapter 3 to create a completely distributed system. A constellation of devices in a sensor network typically produces raw data, which can be stored, annotated and used to train ML models or infer context descriptions, possibly by means of semantic-enhanced ML methods [109]. Such descriptions can act as arguments for constructing and evaluating a BWAF in the Intelligence modules. Cloud nodes could serve as repositories of processed argumentation graphs and ranking outcomes, which can be utilized for additional analysis, statistics, and user-friendly presentation.

4.1 State of the art

In latest years, the argumentation theory originated by Dung [49] has been increasingly applied to IoT scenarios for modeling interactions among smart agents and to enable argumentation-based self-coordination. In [86], argumentation is exploited to coordinate smart vehicles on a congested road. Arguments model both data collected through vehicle devices and possible actions. By processing the argumentation graph, each vehicle agent is able to solve conflicts, identify winning arguments (*i.e.*, suggested actions) and change lane according to the current road configuration. Similarly, in [83] the argumentation graph models object interactions like dialogues in natural language. The main goal –exemplified in case studies on traffic management and ambient-assisted living– is to define an argumentation-based decision-making system, overcoming the limits of conventional rule-based approaches in IoT contexts. Despite both works show the feasibility and usefulness of argumentation in IoT, exploitation of formal argument models and relation evaluation methods are not covered there. A game-theoretic weighted voting scheme is proposed in [61] for conflict resolution in argumentation-based decision making for Social IoT. Each smart object in the environment votes in favor of or against a particular argument in order to find the best possible conclusion. However, this approach lacks the ability for each social object to issue a gradual vote for each identified conclusion, instead of a basic "yes/no" vote.

Combining knowledge representation and reasoning with argumentation frameworks can lead to significant improvements, allowing to handle issues such as defeasibility and inconsistency [102]. Along this vision, early proposals have exploited abstract argumentation for reasoning over inconsistent knowledge bases. In [90], a Generalized Argumentation Framework (GenAF) has been defined to build an argumentation graph upon an underlying KB, where (i) each argumental atom represents a formula in the KB and (ii) each attack relation between two arguments models inconsistency or incoherence between conflicting sources of information. While this work provides a general extension to abstract argumentation, adaptable to different logics for representing knowledge inside arguments, conflict recognition is based only on

consistency check, *i.e.*, satisfiability check on the conjunction of arguments. A fuzzy argument-based classification scheme named CLeFAR has been proposed in [62] for fall detection in ambient-assisted living. By combining a fuzzy reasoning process and the Extended Social Abstract Argumentation (ESAA) framework, CLeFAR verifies whether the event class predicted by a common classifier is correct or not, thus outperforming basic classification schemes. Unfortunately, obtained ontology-based fuzzy arguments are not exploited to evaluate relations between them. A deductive argumentation framework has been proposed in [31] to reason with conflicting and uncertain ontologies. Two different relations underlie the argument structures; unlike [90], each argument is also associated with a weight representing the information certainty degree. In both the above cases, the final argumentation graph is used to study acceptability of arguments, but implementations in concrete scenarios have not been proposed and the reference logical formalisms have not been linked to standard Semantic Web languages, hence a comprehensive working framework is *de-facto* lacking.

As highlighted in [65], knowledge-based approaches can be exploited to: annotate arguments w.r.t. a reference ontology providing the conceptualization and vocabulary for the particular knowledge domain; simplify graph sharing among different agents; autonomously identify support and attack relations among argument descriptions. The present proposal aims to define an ontology-based argumentation framework where DLs are used not only for modeling information, but also to introduce a process –based on semantic matchmaking– identifying and weighting relations through non-standard inferences. Early proposals like DILIGENT [129] acknowledged the potential of OWL-based formal models of argumentation, but cast perplexity on their applicability to a group of human agents. This framework aims to remove this barrier with an approach oriented to software agents.

Another essential aspect that the proposed approach addresses is the argument acceptability process. In applications with a large number of arguments, labeling arguments with the classical *accepted/rejected* evaluation of extension-based semantics is too simplistic and inappropriate for correct decision-making by autonomous agents. In contrast, gradual semantics produces finer assessments of the arguments, based on numerical scores and

rank-order arguments from the most to the least acceptable ones [7]. Furthermore, it can be better suited to pervasive contexts, since even sub-optimal solutions can be considered useful due to unpredictable context changes and the strict computational constraints of devices.

The work in [18] analyzes the literature on the properties of gradual evaluation methods in argumentation. It has identified groups of conceptually related properties, which can be regarded as based on common patterns. In [28] a comparative study on ranking-based semantics has been carried out, underlining their differences in behavior and applicability. Based on a set of 13 well-defined principles that a semantics should satisfy in a bipolar setting, the comparison in [8] has shown extension semantics do not exploit support relations, as they declare all supported and non-supported arguments as equally acceptable. The authors have proposed a novel *exponent-based* semantics for bipolar weighted argumentation, which satisfies all the principles and is not affected by the problem. However, it does not exploit argument relation chains to identify and evaluate supported attacks or indirect attacks; moreover, it only deals with acyclic graphs. The latter issue affects also the gradual BWAF semantics in [97], which has notwithstanding provided useful inspiration for devising the approach proposed in this work. On the other hand, some ranking semantics are able to deal with cyclic weighted argumentative graphs. In [11] three novel semantics are proposed and comprehensively compared with existing semantics defined in literature for evaluating arguments in weighted graphs. Unfortunately, WAFs have some limitations and it is not possible to fully compare them with BWAFs, as (i) attack is the only possible relation between pairs of nodes and (ii) the weight (called *basic score*) is preliminarily assigned to each argument and not to relations, as in the proposed framework. The work in [9] introduces a formalization of the notion of *compensation* in classical Dung-style AF, whereby an argument receiving one strong attack is as good as an argument receiving several weak attacks. A strong attack is made by a non-attacked argument, whereas an attack is deemed weak if it comes from an argument that is attacked by several non-attacked arguments. A large family of semantics allowing compensation, called α-BBS (α Burden-Based Semantics), is formalized.

The role of non-attacked arguments in AFs has been also investigated

in [27], defining six semantics based on the *propagation* of the weight of each argument to its neighbors. The propagation method encompasses two steps: (i) assigning an early weight to each argument so that the weight of unattacked arguments is greater than the attacked ones and (ii) iterative propagation of these weights into the graph, changing their polarities in order to comply with the meaning of attack and defense relations. In [27] propagation occurs with respect to weights defined *a priori* randomly for each argument. On the contrary, in the presented approach all arguments are assigned the same base score, then the strength propagation of paths is calculated by multiplying the weights of all relations in the path. Therefore it actually takes into account both the polarity and the intensity of each relation, which are determined via inferences on the DL concept descriptions representing the arguments. This makes the outcome of strength propagation on each path more clearly interpretable and explainable.

In the extension [29] of the aforementioned work, the authors have introduced a new ranking semantics based on propagation, with an additional parameter to gradually decrease the impact of arguments when the length of the path between two arguments increases. This semantics satisfies two persuasion principles: *procatalepsis*, *i.e.*, it is often effective to anticipate counter-arguments from interlocutors, and *fading*, *i.e.*, long paths of argumentation become ineffective. However, even in that case, propagation takes place with respect to the weights associated to arguments in a preliminary step, by discriminating attacked arguments from unattacked ones.

Ultimately, with respect to the aforesaid literature efforts, the ranking semantics proposed in this framework: (i) is suitable for BWAF; (ii) exploits the propagation of relation weights, formally evaluated in a semantic matchmaking process between pairs of annotations in $\mathcal{ALN}$ DL; (iii) promotes an accurate argument ranking process that satisfies the fading property; (iv) complies with ten of the eleven group properties in [18] for well-defined gradual argumentation semantics.

4.2 Proposed framework

This section describes the proposed deductive argumentative reasoning framework. Semantic Web Languages and non-standard inference services are exploited for the construction, evaluation and explanation of an argumentation graph. The BWAF argumentation model is adopted as reference: bipolar weighted relations between pairs of arguments grant higher flexibility in the evaluation of both relation strength and argument acceptability. Furthermore, a novel propagation-based ranking semantics assesses the acceptability of each argument by assigning a numeric score.

4.2.1 Interpretable Bipolar Weighted Argumentation

Let us consider a multi-agent system and an argumentation graph induced by the semantic annotations exchanged among objects. The argumentation process occurs through the following stages: (i) argument relation appraisal, (ii) argument acceptability evaluation, (iii) explanation of outcomes. The proposed approach exploits non-standard reasoning recalled in Section 2.2.2 to give a structured argument representation to a BWAF. Logic-based explanations concerning the argumentation graph facilitate an interpretable general-purpose argumentation pipeline.

In the directed weighted graph, arguments are knowledge fragments (*i.e.*, semantic annotations) shared by agents. They are expressed as (unfolded and CNF-normalized) $\mathcal{ALN}$ concept expressions w.r.t. a scenario-dependent ontology $\mathcal{T}$. That set of exchanged annotations takes the role of $\mathcal{A}$ in the BWAF $\mathcal{G}$, as recalled in Section 2.3.2, and the set of pairwise agent interactions coincides with the set of relations $\mathcal{R}$ between the corresponding arguments. More formally, given two generic agents A_R and A_S in a network (*e.g.*, smart devices in a pervasive computing environment, supported by the cloud-edge framework described in Chapter 3), their annotations –denoted as R and S respectively– are treated as arguments, since they represent conclusions reached at the end of an internal information processing. If A_S communicates with A_R ($A_S \rightsquigarrow A_R$), the proposed matchmaking-based approach considers R as request and S as resource: inspired by SWoT MAS frameworks such as [108], in fact, S can "respond to needs" expressed in R.

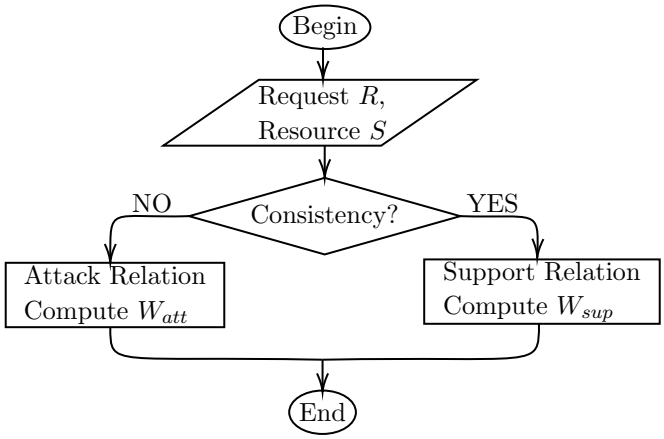


Figure 4.1: Relation type definition algorithm between pairs of arguments

This peculiarity shows a possible argumentative relation can exist with edge orientation from S to R.

Once the direction of a relation is defined, the algorithm illustrated in Figure 4.1 performs the appraisal of its type, discriminating whether S attacks or supports R. A preliminary semantic *consistency check* is performed: if $R \sqcap S$ is satisfiable w.r.t. $\mathcal{T}$, the relation that links S to R is a support, otherwise it is an attack [56]:

$$w_{\mathcal{R}}\left(\langle S, R\rangle\right) = \begin{cases} w_{\mathcal{R},att}\left(\langle S, R\rangle\right) & if\ \mathcal{T} \models R \sqcap S \sqsubseteq \bot \\ w_{\mathcal{R},sup}\left(\langle S, R\rangle\right) & otherwise \end{cases} \tag{4.1}$$

Different strategies are defined for the two cases in Equation 4.1 to properly weigh the $S \rightsquigarrow R$ edge, exploiting non-standard inference services. This process is iterated for each pair of nodes interacting in the graph.

An attack relation exists only if there is inconsistency between the semantic annotations of two arguments. Hence, the following informative contributions are taken into account to compute its weight [56]:

1. the amount of conflicting information between the two arguments;
2. the amount of information confirmed by both arguments;
3. the amount of information in the attacked argument which is neither confirmed nor rebutted by the attacker;

4. any amount of additional information in the attacker which is not present in the attacked argument.

For weighing an attack, Concept Abduction, Contraction and Concept Bonus (Section 2.2.2) can thus be combined in a formula. Each of the above contributions is represented by a signed term of an algebraic sum. Since the weight of an attack must be a real negative value in BWAF, the maximum importance should be given to the term 1, whilst the remaining three contributions intuitively mitigate the attack strength, and therefore they are given a positive sign. This model leads to the following formula:

$$\begin{aligned} w_{\mathcal{R},att}\left(\langle S, R\rangle\right) = &- \alpha\frac{p_c(R,S)}{\|R\|} + \beta\frac{\|Bonus(Bonus(K,S),S)\|}{\|R\|} \\ &+ \gamma\frac{p_a(K,S)}{\|R\|} + \delta\frac{\|Bonus(R,S)\|}{\|R\|\cdot\|S\|} \end{aligned} \tag{4.2}$$

where p_c and p_a are the penalties of Concept Contraction and Abduction (see Section 2.2.2), respectively, and $\|\cdot\|$ is the CNF norm. Considering the matchmaking results and constraining the attack weight in the interval $[-1, 0[$, coefficients have been determined empirically in preliminary tests as: $\alpha = 2.5$, $\beta = 0.1$, $\gamma = 0.05$, and $\delta = 0.05$.

Given an inconsistency between R and S, Concept Contraction penalty $p_c(R, S)$ evaluates the amount of conflicting information, computed as norm of the *Give Up* concept: $p_c(R, S) = \|G\|$. That term is normalized by $\|R\|$, *i.e.*, the worst-case value of $p_c(R, S)$ (all the information in R is rebutted by S). Second term in Equation 4.2 is obtained via a nested pair of Bonuses: the inner $Bonus(K, S)$ gives the additional information in S w.r.t. K (the consistent part of R obtained from Concept Contraction); the outer Bonus identifies what is "not additional", *i.e.*, common to both K and S. The third term comes from the penalty p_a of Concept Abduction between *Keep* K and S: it is computed as $\|H\|$, where the *Hypothesis* H is the part of K not matched by S, and thus the part of R which is neither in conflict nor confirmed w.r.t. S. Both second and third terms are normalized by $\|R\|$, which is the maximum possible value. Finally, the last component in Equation 4.2 quantifies the additional information the attacking argument

S has w.r.t. to the attacked one R; in this case the normalization is by the product of the norms of R and S.

As explained previously, an argument S supports R only if no clash between them exists. Treating R as request and S as resource in typical matchmaking settings, two contributions entail [56]:

1. the amount of information missing in S to reach a full match with R;
2. the amount of additional information S has w.r.t. R.

This model leads to the following formula:

$$w_{\mathcal{R},sup}\left(\langle S,R\rangle\right) = 1 - \frac{p_a(R,S)}{\|R\|}\left(1 - \frac{\|Bonus(R,S)\|}{\|S\|}\right) \tag{4.3}$$

where p_a and Bonus have been already defined. The maximum support of S to R occurs when $\mathcal{T} \models S \sqsubseteq R \Leftrightarrow H \equiv \top \Rightarrow p_a(R,S) = 0$, whereas support is minimum when $H = R \Rightarrow p_a(R,S)/\|R\| = 1$. The last term of Equation 4.3 quantifies the additional information of S w.r.t. R through the Bonus inference, analogously to the Attack formula.

4.2.2 Propagation-based ranking semantics

In highly dynamic and unpredictable scenarios, collaborative autonomous decision-making depends on fine-grained information evaluation. Therefore gradual argument acceptability assessment can be beneficial w.r.t. classical extension-based semantics. This section outlines the novel argumentative ranking semantics, based on path strength propagation in an interpretable BWAF.

Definition 21 (Strength Propagation) *Let $\mathcal{G} = \langle \mathcal{A}, \mathcal{R}, w_{\mathcal{R}} \rangle$ be a BWAF and $x_1, x_n \in \mathcal{A}$ two arguments such that there exists a path $p^* = \langle x_1, x_2, \ldots, x_n \rangle$. The* strength propagation *(sp) from x_1 to x_n for p^* is given by:*

$$sp\left(x_1, x_n\right)_{p^*} = \prod_{i=2}^{n} w_{\mathcal{R}}\left(\langle x_{i-1}, x_i\rangle\right) \tag{4.4}$$

Basically, $sp\left(x_1, x_n\right)_{p^*}$ computes the strength of a path p^* from x_1 to x_n by multiplying the weight of its constituent relations. By construction, $sp\left(\cdot\right)$ is always in $[-1, 0[\cup]0, 1]$ and returns a measure of the role x_1 is playing for x_n: a positive value means a *bw-supports* relation, otherwise a *bw-attacks* one. Each path between any two arguments has its own strength propagation.

In a BWAF, the possibility for an argument to be the target of a relation determines its type. This consideration leads to what follows.

Definition 22 (Connected and free arguments) *Let $\mathcal{G} = \langle \mathcal{A}, \mathcal{R}, w_{\mathcal{R}} \rangle$ be a BWAF. The set of arguments $\mathcal{A}$ is partitioned in two disjoint subsets:*

- $\mathcal{A}^c = \{y \mid y \in \mathcal{A} \ \wedge \exists\ x \in \mathcal{A}$ *s.t.* $\langle x, y \rangle \in \mathcal{R}\}$ *is the subset of arguments which receive at least one attack or support, denoted as* connected *arguments;*
- $\mathcal{A}^f = \mathcal{A} \setminus \mathcal{A}^c$ *is the subset of arguments in $\mathcal{A}$ which are not attacked or supported by any other argument, hereinafter called* free *arguments.*

The proposed ranking semantics is based on the *fading* principle [29], stating that the *length* of a chain of arguments should be inversely proportional to the *impact* of that chain on the final argument. The idea of fading derives from the observation that, in common dialogue, longer chains of arguments become less effective, because short-term memory is limited and people easily lose track of far-reaching consequences of arguments and relations. Similarly, the majority of pervasive computing devices have strictly constrained memory and storage space, therefore fading allows agents to weigh less or "forget" those arguments which are farther in time and/or space. The proposed path strength propagation via multiplication of relation weights achieves this property rather naturally: regardless of polarity, the absolute value of each weight is in $]0, 1]$ and therefore the overall intensity can never increase and will likely decrease for longer paths.

In the proposed BWAF ranking semantics, the acceptability assessment of arguments is an iterative process, outlined in Algorithm 1. It starts after arguments in the graph have been annotated with DL descriptions and relation edges have been directed and weighed as explained in Section 4.2.1.

Require: BWAF $\mathcal{G} = \langle \mathcal{A}, \mathcal{R}, w_{\mathcal{R}} \rangle$, $\zeta \in]0,1[$, $\epsilon \in \mathbb{R}^+$, computation of $\mathcal{A}^f$, $\mathcal{A}^c$
Ensure: Ranking of arguments $R = \{\langle a_i, s_i \rangle\}, a_i \in \mathcal{A}, s_i \in [-1,1]$

$R := \varnothing$
for all $a_i \in \mathcal{A}$ **do**
  if $a_i \in \mathcal{A}^c$ **then**
    $flagStop_{counter}[a_i] := 0$
  end if
  $s_i := 0$
  add $\langle a_i, s_i \rangle$ to R
end for
$fastStop := false$
for $j := 1$, $fastStop \neq true$, $j++$ **do**
  for all $a_i \in \mathcal{A}^c$ **do**
    $X := Y := 0$
    $\langle \mathcal{P}^f_j(a_i), \mathcal{P}^c_j(a_i) \rangle := retrievePaths(a_i, j)$ // retrieve all free-born and connected-born paths with lenght j, ending in a_i, and satisfying constraints (C1) and (C2)
    if $(\mathcal{P}^f_j(a_i) \neq \varnothing)$ **then**
      $X = \frac{\zeta^j}{|\mathcal{P}^f_j(a_i)|} \sum_{p \in \mathcal{P}^f_j(a_i)} sp(\alpha, a_i)$
    end if
    if $(\mathcal{P}^c_j(a_i) \neq \varnothing)$ **then**
      $Y = \frac{(1-\zeta)^j}{|\mathcal{P}^c_j(a_i)|} \sum_{p \in \mathcal{P}^c_j(a_i)} sp(\alpha, a_i)$
    end if
    $rank_{upd} := s_i + X + Y$
    update $\langle a_i, s_i := rank_{upd} \rangle$ in R
    if $(|rank_{upd} - s_i| < \epsilon$ **then**
      $flagStop_{counter}[a_i] ++$
      if $(flagStop_{counter}[a_i] == 2)$ **then**
        remove a_i from $\mathcal{A}^c$
      end if
    else
      $flagStop_{counter}[a_i] := 0$
    end if
    if $(\mathcal{A}^c == \varnothing)$ **then**
      $fastStop := true$
    end if
  end for
end for
$convertRanking(R)$
return R

Algorithm 1: Propagation-based Ranking Semantics

Its outcome consists of a ranking, associating each argument $\theta \in \mathcal{A}$ with an acceptability score consisting in a real number in the $[-1, 1]$ interval, where 0 stands for neutrality, and higher (respectively, lower) values proportionally label acceptable (resp. unacceptable) arguments. At each step $i > 0$, the procedure identifies all paths of length i ending in the argument under evaluation.

Definition 23 (Path length) *Let $\mathcal{G} = \langle \mathcal{A}, \mathcal{R}, w_{\mathcal{R}} \rangle$ be a BWAF and $x_1, x_n \in \mathcal{A}$ two arguments s.t. there exists a path $p^* = \langle x_1, x_2, \ldots, x_n \rangle$. The* length *$i$ of the path p^* is the number $n-1$ of relations $\langle x_{j-1}, x_j \rangle$, where $j = 2, \ldots, n$.*

As per Definition 22, the kind of source node of a path defines its type; consequently, Algorithm 1 evaluates the different contributions separately in computing the acceptability score (lines 14–19).

Definition 24 (Free-born and connected-born paths) *Let $\mathcal{G} = \langle \mathcal{A}, \mathcal{R}, w_{\mathcal{R}} \rangle$ be a BWAF and $\alpha, \theta \in \mathcal{A}$ two arguments. Any path $p^* = \langle \alpha, \ldots, \theta \rangle$ of any length i ending in θ belongs to one of the following two disjoint sets:*

- Free-born *paths $\mathcal{P}_i^f(\theta)$, starting in an argument $\alpha \in \mathcal{A}^f$*
- Connected-born *paths $\mathcal{P}_i^c(\theta)$, starting in an argument $\alpha \in \mathcal{A}^c$*

However, w.r.t. the strength propagation value taken by each path, the sets of free-born (respectively, connected-born) paths is further separable into two disjoint subsets as reported hereafter.

Definition 25 *Let $\mathcal{G} = \langle \mathcal{A}, \mathcal{R}, w_{\mathcal{R}} \rangle$ be a BWAF, $\theta \in \mathcal{A}$, $\mathcal{P}_i^f(\theta)$ and $\mathcal{P}_i^c(\theta)$ the sets of free-born paths and connected-born paths of any length i ending in θ. Each path $p^* \in \mathcal{P}_i^f(\theta)$ can be assigned to:*

- Support Free-born paths *set $\mathcal{P}_i^{f+}(\theta)$, if $sp(\cdot)_{p^*} \in]0, 1]$, or*
- Attack Free-born paths *set $\mathcal{P}_i^{f-}(\theta)$, if $sp(\cdot)_{p^*} \in [-1, 0[$*

Similarly, each path $q^* \in \mathcal{P}_i^c(\theta)$ is in:

- Support Connected-born paths *set $\mathcal{P}_i^{c+}(\theta)$, if $sp(\cdot)_{q^*} \in]0, 1]$, or*

- Attack Connected-born paths *set* $\mathcal{P}_i^{c-}(\theta)$ *if,* $sp\,(\cdot)_{q^*} \in [-1, 0[$

Consequently $\mathcal{P}_i^f(\theta) = \mathcal{P}_i^{f+}(\theta) \cup \mathcal{P}_i^{f-}(\theta)$ *and* $\mathcal{P}_i^c(\theta) = \mathcal{P}_i^{c+}(\theta) \cup \mathcal{P}_i^{c-}(\theta)$.

The iterative ranking procedure initially assumes neutral acceptability for all arguments: $\forall\ \theta \in \mathcal{A} : s_0(\theta) = 0$ (lines 6–7 of Algorithm 1). This is also the final score for free arguments, while for each connected argument $\theta \in \mathcal{A}^c$ at each step $i > 0$, lines 11–34 compute:

$$s_i(\theta) = s_{i-1}(\theta) + X + Y \tag{4.5}$$

where:

$$X = \begin{cases} \frac{\zeta^i}{|\mathcal{P}_i^f(\theta)|}\left(|\mathcal{P}_i^{f+}(\theta)| \sum_{p \in \mathcal{P}_i^{f+}(\theta)} sp(\alpha_p, \theta) + |\mathcal{P}_i^{f-}(\theta)| \sum_{p \in \mathcal{P}_i^{f-}(\theta)} sp(\alpha_p, \theta)\right) & if\ \mathcal{P}_i^f(\theta) \neq \varnothing \\ 0 & otherwise \end{cases}$$

$$Y = \begin{cases} \frac{(1-\zeta)^i}{|\mathcal{P}_i^c(\theta)|}\left(|\mathcal{P}_i^{c+}(\theta)| \sum_{p \in \mathcal{P}_i^{c+}(\theta)} sp(\alpha_p, \theta) + |\mathcal{P}_i^{c-}(\theta)| \sum_{p \in \mathcal{P}_i^{c-}(\theta)} sp(\alpha_p, \theta)\right) & if\ \mathcal{P}_i^c(\theta) \neq \varnothing \\ 0 & otherwise \end{cases}$$

with $sp(\alpha_p, \theta)$ evaluated for the path starting in argument α_p and ending in argument θ as per Equation 4.4 and $0 < \zeta < 1$ a coefficient (empirically set to 0.8 after preliminary tests) having two purposes: satisfy the fading principle (by means of the exponential) and give greater weight to free-born paths. The latter idea is justified by the *void precedence* property in ranking semantics [7], stating that non-attacked arguments should have a higher acceptability rank than attacked ones, and also bears some similarity to the *evidential* interpretation of support [93], as free-born arguments include *prima facie* arguments as defined in that context.

In order to ensure termination and make the algorithm less computationally expensive, two constraints are imposed in identifying the paths (with increasing length i) suitable to update the acceptability score of an argument θ. For each step $i \in \mathbb{N}$, $i > 0$, any path $p^* = \langle x_1, x_2, \ldots, x_i, x_{i+1} = \theta \rangle$ with length i is admissible for computation only if both the following constraints are satisfied:

(C1) the source argument x_1 and any other argument x_k, with $k = 2, 3, \ldots, i$

are different from the argument $x_{i+1} = \theta$ whose acceptability is being evaluated;

(C2) for each argument a repeated $n \geq 3$ times in the path p^*, all the $n-1$ sequences of arguments included between pairs of occurrences of a are different from each other.

Constraint $(C1)$ is imposed because (i) the sp of any path originating from an argument θ must not impact the computation of the acceptability rank of the same argument θ, and, (ii) if the same θ is in a position $k < i+1$ of path p^*, it is plausible that the contribution of the path to the rank of θ has already been considered in one of the previous iterations. Constraint $(C2)$ prevents infinite loops, instead, by discarding paths with repeated sub-paths.

In addition to path admissibility, in order to prevent non-termination, lines 22–29 of Algorithm 1 further check the acceptability score. In fact, if the absolute value of the difference between $s_{i+1}(\theta)$ and $s_i(\theta)$ is less than a constant ϵ for two consecutive iterations, score convergence is accepted and the procedure stops. Preliminary tests have found the value $\epsilon = 0.05$ consistent with this purpose. In Algorithm 1, following a preliminary initialization in lines 3–5, structure $flagStop_{counter}$ keeps for each connected argument how many times the absolute value of the difference between two consecutive scores is less than ϵ. If for a connected argument a_i the value $flagStop_{counter}[a_i]$ is equal to 2, the argument a_i is removed from the set $\mathcal{A}^c$, since the convergence of the acceptability score has been reached and no iteration is needed anymore (lines 24–26).

At the end, to ensure the convergence of the acceptability scores of each argument of a particular BWAF graph in the interval [-1,1] and to extract quantitative information on the acceptability of the arguments from the ranking, the *convertRanking* function is adopted (line 35). Let f be the logistic function and $x = s_i$ the score of each argument a_i computed by the algorithm; then

$$convertRanking(x) = 2f(kx) - 1 \tag{4.6}$$

with k sigmoid smoothing factor, which has been set to 3 after preliminary tests.

The proposed ranking semantics complies with all the *group properties* (GPs) identified in [18], except GP9. GPs identify groups of conceptually related properties in literature, which can be regarded as based on common patterns for gradual argumentation. The proposed approach does not comply with GP9 (*"A higher base score gives a higher strength"* [18]) by design, since the base score is equal for all the arguments (line 6 of Algorithm 1) in the bipolar weighted graph.

It should be noted that, in this approach, the number of required iterations to compute the acceptability score is not the same for all arguments in a BWAF: the early score for free arguments is 0 and no further calculations are needed; conversely, for each connected argument the number of iterations depends on the number, length and *sp* of the paths ending in it.

Finally, the proposed semantics induces a gradual acceptability ranking of arguments in a BWAF.

Property 1 *The propagation-based semantics* $\mathcal{S}(\cdot)$ *associates a ranking* $\succeq_{\mathcal{G}}^{\mathcal{S}}$ *on* $\mathcal{A}$ *to any BWAF* $\mathcal{G}$*, such that* $\forall\ a, b \in \mathcal{A}$*,* $a \succeq_{\mathcal{G}}^{\mathcal{S}} b$ *iff* $\mathcal{S}(a) \geq \mathcal{S}(b)$*.*

Overall, the ranking procedure ensures a gradual assessment of the acceptability of the arguments in a BWAF. Main features of the adoption of this semantics can be summarized as:

- the ability to manage bipolar weighted graphs with cycles and paths of any length;
- completeness of the algorithm thanks to path admissibility constraints and convergence condition;
- persistence of the value 0 as acceptability threshold as well as the score of free arguments;
- convergence of the acceptability scores of the connected arguments in the interval $[-1, 1]$ and correlation between the numerical value and the acceptability of the argument *i.e.*, the closer to 1 the score of an argument, the more acceptable it is;
- compliance with the fading principle, as the strength for longer paths is dampened.

Compared to *Discussion-based semantics* (Dbs) [7], which basically counts the number of linear discussions (LDs) of an argument as the LD length increases, the proposed approach computes the weighted contributions *sp* of the individual paths at each iteration and discriminates free-born from connected-born paths. The *Burden-based semantics* (Bbs) [7] elaborates the ranking by comparing the arguments lexicographically, based on their burden numbers, and the update of the burden numbers occurs considering all the attack relationships between arguments in an AF as equivalent. Unlike the semantics presented, both Dbs and Bbs are unable to solve graphs with cycles, as recalled in Section 2.3.1.

The above framework is general-purpose, while striving for an efficient usage of computational resources, which is essential particularly for pervasive computing agents.

4.3 Case study

In order to validate the proposal and highlight its characteristics, a case study has been conducted in the real-time strategy game *StarCraft II* (SC2). It is a competitive MAS which has been chosen because its complexity and fast dynamics have significant similarities to pervasive computing contexts. The (player) agent is constantly interacting with the (game) environment, receiving stimuli and performing actions; the environment cannot be considered merely as a setting or substrate for the agents' decisions and actions, but it is a first-class entity in the MAS [137].

Figure 4.2 shows the proposed deductive argumentative agent architecture. The strategic player agent is made up of five main blocks interacting in a continuous loop with SC2:

- *Field Data Collection:* gathers real-time game state features from a running SC2 instance exploiting the *StarCraft II API*[1];
- *Semantic Annotation:* performs semantic enrichment of raw data with respect to a reference DL domain ontology, producing annotated concept descriptions;

[1]StarCraft II API: `https://github.com/Blizzard/s2client-proto`

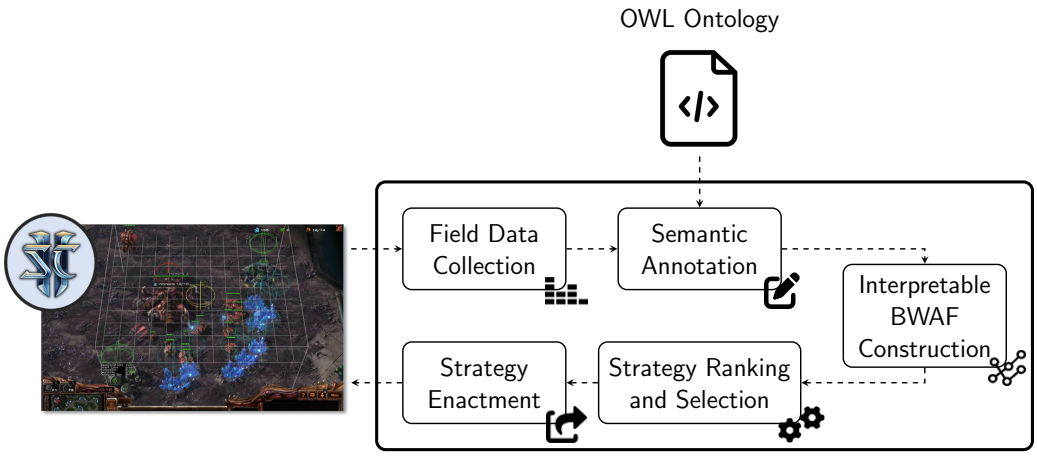


Figure 4.2: Deductive argumentative agent architecture

- *Interpretable BWAF Construction:* adopts the semantic annotations as arguments in a bipolar weighted graph, and evaluates the type and weight of relations between arguments by exploiting non-standard inference services for semantic matchmaking, as explained in Section 4.2.1;

- *Strategy Ranking and Selection:* performs a context-dependent acceptability ranking of the arguments in the graph, by means of the strength propagation semantics described in Section 4.2.2. This step allows to decide the agent plan among all arguments representing possible strategies;

- *Strategy Enactment*: encodes the semantic annotation representing the selected strategy into a sequence of game commands to be applied through the SC2 environment interface.

For the sake of clarity, a simple example of skirmish match is now described among allied units (*Player 1*), controlled by the argumentative deductive reasoning agent, and enemy units (*Player 2*). Since a match ends when all the units of one of the two players are destroyed, it is essential for a player to identify the best assault and defense strategies to win.

In an asymmetrical battle scenario, 6 *Marine units of the Terran enemy faction are exerting an assault on* 3 *Zealot units of the Protoss allied army. Although outnumbered, the allied army has better health conditions than the*

Figure 4.3: SC2 case study scenario

opponent. Figure 4.3 depicts the situation.

The case study aims to show the argumentation process for selecting the best strategy in a given instant of the battle. Each possible plan is described via a semantic annotation providing information about both its type and game conditions which make it suitable. The StarCraft II battle state features which have been taken into account include: types and number of involved units; differences in assault range, movement speed, and armor between the two brigades; type of assault suffered by the player agent's army; Hit Points (HPs) and shields availability levels of the allied and enemy units. In this setting, let us consider the following five strategies:

1. *Healthiest Assault* (HA): assault the healthiest enemy unit;
2. *Nearest Assault* (NUA): assault the nearest enemy unit;
3. *Least Healthy Assault* (LHA): assault the enemy unit with the lowest health level;
4. *Least Healthy Defense* (LHD): protect the agent's unit with the lowest

health level, by adopting a compact formation and counter-assaulting the nearest enemy unit assaulting it;

5. *Kiting* (KT): defend by distancing enemy units, as the assault range of the agent's units is longer, so they can hit without being counter-assaulted.

The aforementioned battle plans are grouped in two families: the first three strategies are *Assaults* and the remaining ones are *Defenses*. This strategy hierarchy is useful for dividing the decision-making process in two subsequent steps: determine first the most promising type of approach to the enemy in the current context, and subsequently select the most proper strategy among those in the selected category. This stratification decreases the number of alternatives to assess in each step, thereby reducing the size of the argumentation graph and leading to computational cost savings. Validating this kind of hierarchical decision-making approach is also useful to make the whole framework more easily transferable to scenarios where distributed computing devices (i) are resource-constrained in terms of processing and memory capabilities and (ii) can be grouped in clusters by locality.

For the case study an OWL 2 Knowledge Base has been modeled in the moderately expressive $\mathcal{ALN}$ DL: the language choice depends on the non-standard inferences explained in Section 2.2.2. An acyclic ontology $\mathcal{T}$ contains the domain conceptualization along specific patterns. Figure 4.4 reports an excerpt of the modeled SC2 ontology exploited by the knowledge-based agent: for each feature (*e.g.*, units numerousness, hit points, armor, attack range, damage per second, shields), $\mathcal{T}$ includes a partonomy, *i.e.*, each parameter is modeled by means of a taxonomy of concepts (each one with its own properties) representing all significant configurations or value ranges it may take in the domain of interest. Therefore, all modeled classes describe the game state features the player agent can sense or act upon through the game API. For the sake of simplicity, the example focuses on units numerousness, health and tactical conditions of ally and enemy units, and enemy units position and formation. They are information fragments gathered from SC2 API-based probes equivalent to sensing peripherals in scenarios involving agents embodied in smart devices. Furthermore, each

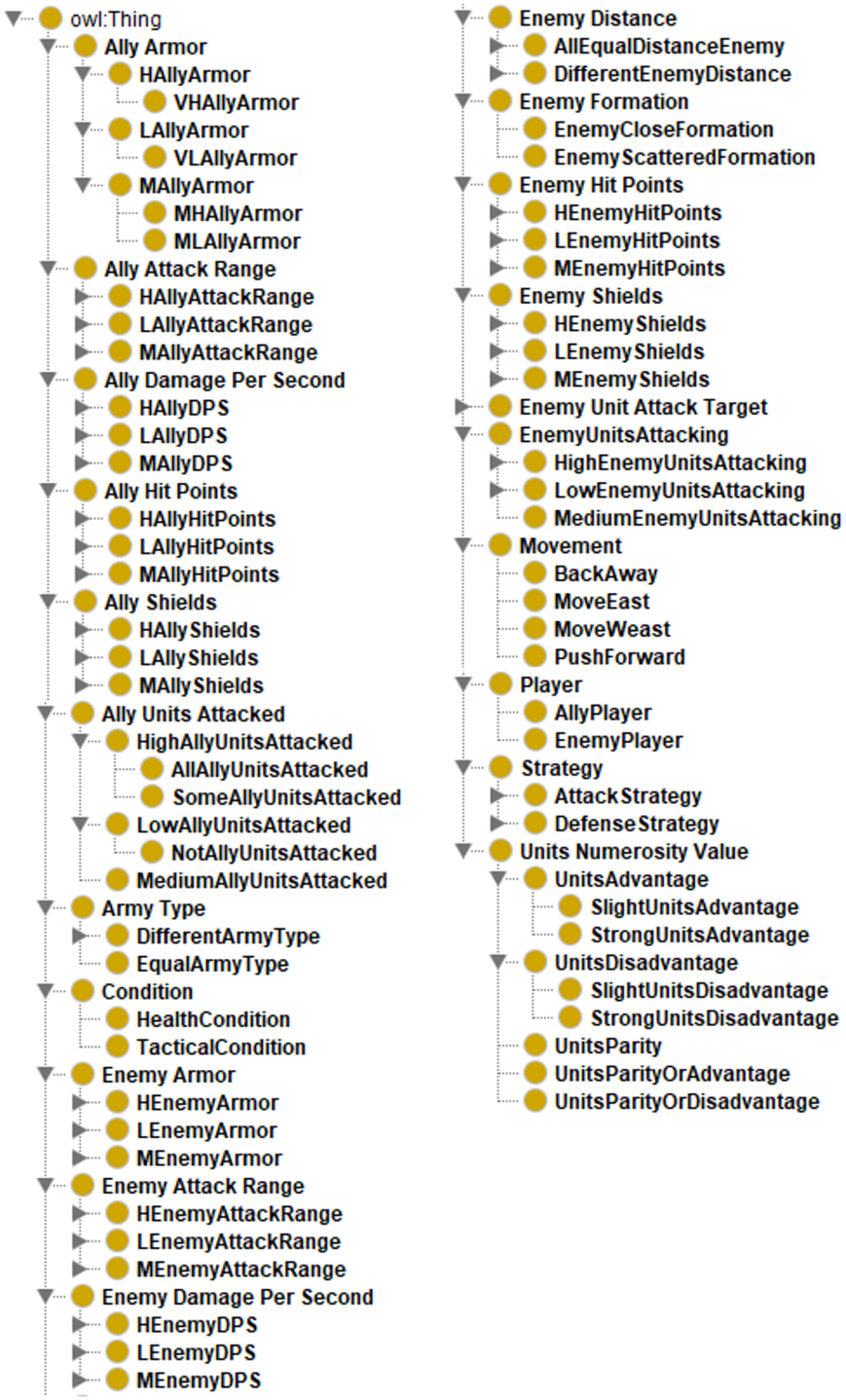


Figure 4.4: Ontology excerpt for the StarCraft II agent

available strategy and relevant in-game condition is modeled as an individual of the ABox and is used as a single argument in the BWAF. The Tiny-ME reasoner [105] has been leveraged to identify type and weight of interactions between pairs of arguments, as discussed in Section 4.2.1.

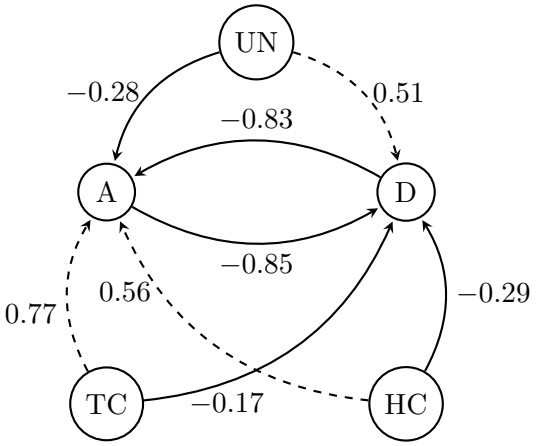


Figure 4.5: $\mathcal{G}_1$: BWAF of the first deductive argumentative stage

Based on the scenario depicted in Figure 4.3, the BWAF $\mathcal{G}_1$ shown in Figure 4.5 is generated in order to select the most appropriate family of strategies. The node describing the *Assault* (A) category is supported by: (i) *Health Condition* (HC), as ally units have high HPs and medium amount of shields whereas enemy units have medium HPs and very low shield levels; and (ii) *Tactical Condition* (TC), as the allied player has a medium amount of armor and, due to their intrinsic characteristics, its units have higher attack range and Damage Per Second (DPS) than the enemy units. On the other hand, argument A is attacked by *Units Numerousness* (UN), since allied units are outnumbered, a fact which discourages offensive tactics. Dually, the *Defense* (D) strategies are attacked by HC and TC, as a protective battle plan is not completely justified if allied units are mostly healthy and the enemy army has low armor values; conversely, *Defense* is supported by UN, due to the numerical inferiority of allied units. Finally arguments A and D attack each other, as they are mutually exclusive. To clarify and illustrate how argument relations are assessed, an example referred to the evaluation of the relation between UN and A is detailed. Arguments are reported in OWL 2 *Manchester Syntax* [66] for the sake of readability.

```
A: AttackStrategy and (hasAllyUnitsAttacked some owl:Thing) and
(hasAllyUnitsAttacked only LowAllyUnitsAttacked) and
(hasUnitsNumerosity some owl:Thing) and (hasUnitsNumerosity only
UnitsParityOrAdvantage) and (hasAllyArmor some owl:Thing) and
(hasAllyArmor only MHAllyArmor) and (hasAllyHitPoints some owl:Thing) and
(hasAllyHitPoints only HAllyHitPoints) and (hasEnemyArmor some owl:Thing)
and (hasEnemyArmor only LEnemyArmor) and (hasEnemyHitPoints some
owl:Thing) and (hasEnemyHitPoints only MLEnemyHitPoints) and
(hasEnemyShields some owl:Thing) and (hasEnemyShields only LEnemyShields)
and (hasEnemyUnitsAttacking some owl:Thing) and (hasEnemyUnitsAttacking
only LowEnemyUnitsAttacking)
```

```
UN: AllyPlayer and (hasAllyUnitsAttacked some owl:Thing) and
(hasAllyUnitsAttacked only AllAllyUnitsAttacked) and
(hasUnitsNumerosity some owl:Thing) and (hasUnitsNumerosity only
UnitsDisadvantage) and(hasEnemyUnitsAttacking some owl:Thing) and
(hasEnemyUnitsAttacking only AllEnemyUnitsAttacking)
```

A and *UN* play the roles of request and resource, respectively, in the algorithm in Figure 4.1 used to identify the type of relation as an attack or a support. The preliminary consistency check fails, as the conjunction of the two expressions is unsatisfiable due to the following clashes of disjoint classes occurring as fillers of the same properties: (i) *LowAllyUnitsAttacked* and *AllAllyUnitsAttacked*; (ii) *UnitsParityOrAdvantage* and *UnitsDisadvantage* and (iii) *LowEnemyUnitsAttacking* and *AllEnemyUnitsAttacking*. The relation is thus recognized as an attack. Concept Contraction is invoked to highlight two OWL class expressions corresponding to the conflicting requirements *G* and the compatible version *K* of the argument *A* w.r.t. the annotation of *UN*.

```
G := hasAllyUnitsAttacked only (LowAllyUnitsAttacked and
(not HighAllyUnitsAttacked)) and hasUnitsNumerosity only
(UnitsParityOrAdvantage and (not UnitsDisadvantage)) and
hasEnemyUnitsAttacking only (LowEnemyUnitsAttacking and
(not HighEnemyUnitsAttacking))
```

```
K := AttackStrategy and (hasAllyArmor only (MAllyArmor and MHAllyArmor))
```

```
and (hasAllyHitPoints only HAllyHitPoints) and (hasEnemyArmor only
LEnemyArmor) and (hasEnemyHitPoints only (MEnemyHitPoints and
MLEnemyHitPoints)) and (hasEnemyShields only LEnemyShields) and
(hasAllyUnitsAttacked some owl:Thing) and (hasUnitsNumerosity some
owl:Thing) and (hasAllyArmor some owl:Thing) and (hasAllyHitPoints some
owl:Thing) and (hasEnemyArmor some owl:Thing) and (hasEnemyHitPoints
some owl:Thing) and (hasEnemyShields some owl:Thing) and
(hasEnemyUnitsAttacking some owl:Thing)
```

The attack weight is computed according to Equation 4.2. The first term of the formula expresses the strength of conflict between the arguments: $-\frac{p_c(A,UN)}{||A||} = -\frac{||G||}{||A||} = -0.127$. The remaining three terms in Equation 4.2, reported with a positive sign, mitigate the attack. Concept Bonus is used twice to define the additional information BB provided by a resource respect to the request. Obtained expressions are reported as follows:

```
Bonus(K,UN) := AllyPlayer and (hasAllyUnitsAttacked only
AllAllyUnitsAttacked and HighAllyUnitsAttacked) and (hasUnitsNumerosity only
UnitsDisadvantage) and (hasEnemyUnitsAttacking only (AllEnemyUnitsAttacking
and HighEnemyUnitsAttacking))
BB := Bonus(Bonus(K,UN),UN) := (hasAllyUnitsAttacked some owl:Thing) and
(hasUnitsNumerosity some owl:Thing) and (hasEnemyUnitsAttacking some owl:Thing)
```

In particular, the second term quantifies the common elements between the two arguments: $\frac{||BB||}{||A||} = 0.064$. The third term of the formula reflects the information defined in the attacked argument A but missing in the attacking node UN. It is obtained as the Hypothesis (H) expression computed by the Concept Abduction inference service between K and UN:

```
H := AttackStrategy and (hasAllyArmor only (MAllyArmor and MHAllyArmor))
and (hasAllyHitPoints only HAllyHitPoints) and (hasEnemyArmor only
LEnemyArmor) and (hasEnemyHitPoints only (MEnemyHitPoints and
MLEnemyHitPoints)) and (hasEnemyShields only LEnemyShields) and
(hasAllyArmor some owl:Thing) and (hasAllyHitPoints some owl:Thing) and
```

```
(hasEnemyArmor some owl:Thing) and (hasEnemyHitPoints some owl:Thing)
and (hasEnemyShields some owl:Thing)
```

The related weight can be expressed as: $\frac{p_a(K,UN)}{||A||} = \frac{||H||}{||A||} = 0.702$. Finally, the information B provided by the attacking argument UN in addition to the description of the attacked node A is determined as:

```
B := Bonus(A,UN) := AllyPlayer and (hasAllyUnitsAttacked only
(AllAllyUnitsAttacked and HighAllyUnitsAttacked)) and (hasUnitsNumerosity only
UnitsDisadvantage) and (hasEnemyUnitsAttacking only (AllEnemyUnitsAttacking
and HighEnemyUnitsAttacking))
```

This term corresponds to $\frac{||B||}{||A||\cdot||\text{UN}||} = 0.013$. The weighted algebraic sum of the above terms is -0.28, corresponding to the value associated to the attack edge from UN to A in the BWAF $\mathcal{G}_1$ depicted in Figure 4.5. In the proposed framework, concept expressions G, BB, H, and B represent the knowledge-based annotations providing the formal explanation of the BWAF and making the obtained results more human- and machine-understandable.

Table 4.1: Acceptability Ranking on $\mathcal{G}_1$

Step i	***s(UN)***	***s(A)***	***s(D)***	***s(TC)***	***s(HC)***
0	0	0	0	0	0
1		0.467	-0.277		
2		0.538	-0.704		
final	**0**	**0.668**	**-0.784**	**0**	**0**

All the edges in graph $\mathcal{G}_1$ are evaluated following the same approach. Then, the proposed ranking semantics is executed. Table 4.1 shows the acceptability scores for each argument; shaded cells indicate the final values obtained by means of the *convertRanking* function described in Section 4.2.2. After the initialization step ($i = 0$) performed for all arguments, no subsequent iteration is needed for *s(UN)*, *s(TC)* and *s(HC)*, as they are free arguments. Instead, the processing goes on for A and D. Let us focus on argument A; for $i = 1$, paths ending in A are $\langle \text{UN}, A\rangle$, $\langle TC, A\rangle$, $\langle HC, A\rangle$ and $\langle D, A\rangle$. The first, second and third path belong to $\mathcal{P}_1^f(A)$, whereas the last

one is in $\mathcal{P}_1^c(A)$. The *sp* values of the four paths (as arranged in Equation 4.5) contribute to $s_1(A) = 0.467$. In the subsequent iteration ($i = 2$), the paths ending in A are $\langle \text{UN}, D, A\rangle$, $\langle TC, D, A\rangle$ and $\langle HC, D, A\rangle$, which contribute to $s_2(A) = 0.538$. The algorithm for calculating the acceptability score of A ends at $i = 2$ because the only path of length 3 $\langle D, A, D, A\rangle$ violates the admissibility constraint of no repeated sub-path. The *final* values in Table 4.1 are obtained by appyling the *convertRanking* function in Equation 4.6. The execution of the proposed ranking semantics for $\mathcal{G}_1$ produces the following results:
$s(\texttt{A}) = 0.668 > s(\texttt{UN}) = s(\texttt{TC}) = s(\texttt{HC}) = 0.0 > s(\texttt{D}) = -0.784$.

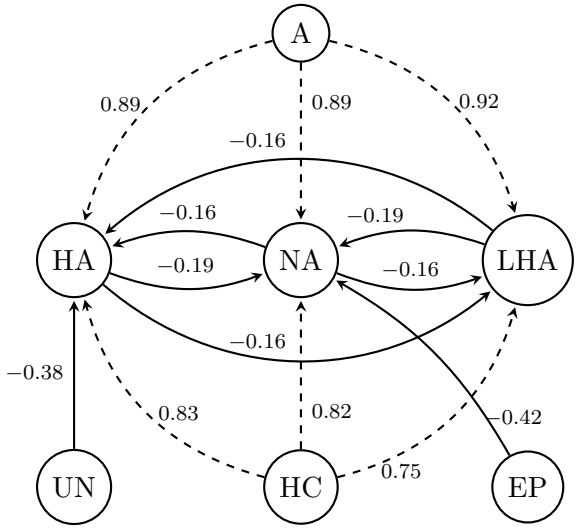


Figure 4.6: $\mathcal{G}_2$: BWAF of the second deductive argumentative stage

The *Assault* argument wins the argumentative dispute: in the current state of the game an offensive strategy is to be preferred. This enables the second argumentation stage, based on the BWAF graph $\mathcal{G}_2$ in Figure 4.6, aiming to identify the most proper battle strategy in the *Assault* family. The three strategies *Healthiest Assault* (*HA*), *Nearest Assault* (*NA*) and *Least Healthy Assault* (*LHA*) are all supported by the generic *Assault* node and have mutual attack relations, since only one can be selected. In addition, *HA* is attacked by *UN*, which recommends this strategy only when the number of allied units is greater than enemy ones. Furthermore, the node *Enemy Position* (*EP*), characterized by equal distance of all enemy units to

Table 4.2: Acceptability Ranking on $\mathcal{G}_2$

Step i	***s(A)***	***s(NA)***	***s(HA)***	***s(LHA)***	***s(UN)*** ***s(EP)*** ***s(HC)***
0	0	0	0	0	0
1		0.730	0.754	1.277	
2		0.400	0.490	1.074	
3		0.442	0.528	1.103	
final	**0**	**0.580**	**0.659**	**0.929**	**0**

allied ones and a scattered formation, attacks the node *NA*, which requires some enemy units at close range and a compact formation. Conversely, all three strategies are supported by *HC*, as they are suitable in case the allied player has high HP and armor levels. Semantic matchmaking, however, yields slightly different support strengths for the three strategies (0.83, 0.82 and 0.75 respectively), implying they are not completely equivalent from the *HC* viewpoint.

Table 4.2 summarizes the ranking iteration for arguments in $\mathcal{G}_2$. The final outcome is:
$s(\texttt{LHA}) = 0.929 > s(\texttt{HA}) = 0.659 > s(\texttt{NA}) = 0.580 > s(\texttt{A}) = s(\texttt{UN}) = s(\texttt{EP}) = s(\texttt{HC}) = 0.0$.
All the above strategies have a score > 0 and are judged as acceptable in the particular battle situation, but the proposed ranking semantics allows to arrange the strategy arguments w.r.t. effectiveness in a given game context. Selecting the *Least Healthy Assault* among the available *Assault* strategies appears consistent with a qualitative evaluation of the scenario, as it allows Player 1 to quickly eliminate the least healthy opponent units and recover from the numerosity disadvantage.

4.4 Experiments

An experimental campaign has been carried out to evaluate the proposed framework in terms of: (i) efficiency of the strategic decision-making agent integrated in SC2, with performance metrics collected during the execution of

the case study scenario; (ii) performance of the ranking semantics algorithm on large-scale graphs, with tests on different random bipolar weighted graph configurations; (iii) feasibility of argumentation graph pruning w.r.t. both result stability and performance improvement. All tests have been run on a desktop PC equipped with Intel Core i7-4790 CPU (3.6 GHz), 16 GB RAM, and Windows 10 (64 bit) operating system.

Strategic agent efficiency

The first performance evaluation of the propagation-based ranking semantics has been carried out on the $\mathcal{G}_1$ and $\mathcal{G}_2$ graphs reported in Figure 4.5 and 4.6. Table 4.3 includes main graph characteristics (*i.e.*, number of vertexes and edges) and obtained results. All performance results are the average of five cold runs. Columns *T1* and *T2* represent the average turnaround time to construct an interpretable BWAF –as described in Section 4.2.1– and execute the propagation-based ranking semantics (Algorithm 1), respectively. The last column indicates the number of iterations required by Algorithm 1 to compute the acceptability of all arguments. By summing all four time values in Table 4.3, one can notice how a complete strategy evaluation round, comprising the two hierarchical stages with graphs like in Section 4.3, would take $\approx$ 11 ms on average, which allows for fast strategy update against dynamic context conditions in an RTS game: as a reference, SC2 updates the simulation 16 or 22.4 times per second, depending on the game speed setting [136], which corresponds to periods of 62.5 or 44.6 ms, respectively, for the agent's event loop. The processing time of the proposed solution appears appropriate and compatible with the strict SC2 time constraints. The short duration of processing tasks has not allowed a reliable memory consumption measurement, but it is possible to notice that the performance metrics of an agent usually vary by 1 or 2 orders of magnitude if the execution takes place on a mobile or embedded device w.r.t. a PC [113]. Consequently, the obtained results indicate as the proposed argumentation framework appears to be feasible also in pervasive scenarios exploiting resource-constrained devices.

Table 4.3: Performance results on $\mathcal{G}_1$ and $\mathcal{G}_2$

	Vertexes	**Edges**	**T1 (ms)**	**T2 (ms)**	**Iterations**
$\mathcal{G}_1$	5	8	2.421	0.946	2
$\mathcal{G}_2$	7	14	2.347	5.127	3

Ranking semantics performance

In order to provide a more complete performance analysis of the devised ranking semantics, the *Boost Graph Library* (BGL)[2] has been used in the second test session to build random bipolar weighted graphs, where edge weights have a uniform distribution probability in $[-1, 0[\cap]0, 1]$. BGL is a standardized generic interface for composing and browsing graphs that exposes their structure while concealing implementation information.

Small (S), *Medium* (M), and *Large* (L) scale configurations have been generated to measure turnaround time and memory consumption peak, both sensitive metrics in pervasive computing contexts. The provided configurations depend on three parameters:

- number of nodes in the graph: 10, 50, and 100, respectively for S, M and L;
- number of weighted edges: 15 and 30 for S, 200 and 700 for M, 400 and 1500 for L;
- $\langle$Att %, Sup %$\rangle$ pairs of percentages of mutual attack and mutual support relations in the graph for S, M, L: $\langle$0%, 0%$\rangle$, $\langle$20%, 0%$\rangle$, $\langle$40%, 0%$\rangle$, $\langle$0%, 20%$\rangle$, $\langle$0%, 40%$\rangle$, $\langle$20%, 20%$\rangle$.

Each graph configuration test has been run five consecutive times, and the results are the average of the last four runs. The following measurements have been gathered: (i) memory usage peak; (ii) processing time to rank the graph arguments; (iii) iterations required to find a solution. The graphs of every configuration are generated randomly, by setting up the aforementioned parameters only.

[2]Boost Graph Library v1.80.0: `https://www.boost.org/doc/libs/1_80_0/libs/graph/doc/index.html`

Table 4.4: Argumentative graph performance results

Vertexes	**Edges**	**Mutual attack**	**Mutual support**	**Memory (MB)**	**Time (s)**	**Iterations**
10	15	0%	0%	N.A.	0.007	5
10	15	20%	0%	N.A.	0.005	5.4
10	15	40%	0%	N.A.	0.007	6
10	15	0%	20%	N.A.	0.007	6
10	15	0%	40%	N.A.	0.007	6
10	15	20%	20%	N.A.	0.009	6
10	30	0%	0%	N.A.	0.033	3.8
10	30	20%	0%	N.A.	0.023	5
10	30	40%	0%	N.A.	0.033	5
10	30	0%	20%	N.A.	0.040	5.8
10	30	0%	40%	N.A.	0.039	6
10	30	20%	20%	N.A.	0.030	6
50	200	0%	0%	1.6	3.105	6
50	200	20%	0%	2.4	3.459	6
50	200	40%	0%	1.4	3.091	6.6
50	200	0%	20%	1.4	2.353	7
50	200	0%	40%	1.8	3.232	7
50	200	20%	20%	1.2	1.843	7
50	700	0%	0%	26.8	39.988	3
50	700	20%	0%	27.2	42.323	3
50	700	40%	0%	18.6	40.414	3
50	700	0%	20%	13.4	39.716	3
50	700	0%	40%	13.0	40.060	3
50	700	20%	20%	5.4	39.674	3
100	400	0%	0%	3.8	10.401	6
100	400	20%	0%	5.0	15.337	6
100	400	40%	0%	2.8	8.967	6
100	400	0%	20%	3.6	13.600	6
100	400	0%	40%	3.6	10.262	7
100	400	20%	20%	3.2	12.258	7
100	1500	0%	0%	71.0	217.355	3
100	1500	20%	0%	69.6	212.544	3
100	1500	40%	0%	64.0	208.791	3
100	1500	0%	20%	69.6	214.520	3
100	1500	0%	40%	37.0	208.801	3
100	1500	20%	20%	33.0	196.778	3

Experimental results are reported in Table 4.4: short execution time has not allowed to evaluate the memory consumption in Small configurations. Especially in the M and L configurations, required computational resources seem strongly dependent on the number of edges, if the number of vertexes in a graph remains the same. For example, with 50 vertexes, the configuration with 700 edges has taken about half the iterations but 1 order of magnitude higher memory and time than the configuration with 200 relations. Furthermore, as the ⟨Att%, Sup%⟩ pair increases with the same number of nodes and arcs in a graph, the execution time remains almost the same and the use of memory is reduced. In all configurations, of the six hypothesized mutual attack and support couples, the pair (20%, 20%) has the lowest memory usage. This happens because multiple paths quickly satisfy the constraints of Algorithm 1 and it is not necessary to keep them in memory for subsequent iterations. In any case, both medium and large scale configurations appear as unsuitable for real-time applications like SC2 or pervasive computing contexts. These findings limit the amount of knowledge fragments each agent is capable of managing, calling for distributed computation of the argumentative graph.

Table 4.5: Bipolar weighted graph configurations

Configuration	**Vertexes**	**Edges**	**Mutual Relation**
A	100	500	0
B	100	500	20%
C	100	500	40%

Table 4.6: Performance on complete graphs

Configuration	**Time** [s]		**Memory Usage** [MB]	
	μ	σ	μ	σ
A	152.91	251.14	147.94	309.51
B	127.10	213.55	132.20	277.75
C	70.59	125.28	43.82	101.34

Graph pruning

An optimization to reduce computations for Algorithm 1 could consist in pruning all edges having a weight whose absolute value is lower than a threshold Θ from an original argumentation graph $\mathcal{G}$, obtaining a simplified graph $\mathcal{G}'$. Intuitively, edges labeled with smaller absolute values represent argument relations with lower relevance and impact for a particular argumentation problem. However, since the removal of an edge can change paths ending in graph nodes, the overall ranking semantics outcomes for $\mathcal{G}'$ could end up as being significantly different w.r.t. $\mathcal{G}$. In order to investigate the impact of graph pruning on result stability and computational performance (processing time and memory usage peak), tests have been conducted on pruned graphs (PGs) for increasing values of Θ.

Three graph configurations have been used in this experimental campaign. As reported in Table 4.5, the configurations depend on three parameters: (i) number of nodes in the graph; (ii) number of weighted edges; and (iii) percentage of mutual (attack and/or support, chosen randomly) relations, *i.e.*, direct cycles in the graph. The last parameter has been considered in order to evaluate how much the presence of cycles in a BWAF graph can affect the performance of the proposed ranking algorithm. For each configuration, 50 different graphs have been randomly created and tested. Mean μ and standard deviation σ of the following measurements have been gathered: (i) processing time required to rank the graph arguments; (ii) memory usage peak. Also in this case, all tests have been executed five consecutive times and results reported in Table 4.6 correspond to the average of the last four runs. Differences in mutual relations percentage have a clear impact: configuration C with 40% of mutual relations requires less memory and time than configurations A and B. This evidence shows the peculiarity of the proposed algorithm to quickly identify and manage cycles in order to prevent the computation of long and possibly trivial chains of arguments, in full compliance with the fading principle.

Afterwards, for each configuration, starting from an initial graph G1 with $N = 100$ vertexes, the pruning activity has taken place by setting three different values for the threshold (Θ), obtaining PGs as follows: G2, for $\Theta =$

Table 4.7: Performance metrics for total ranking

Config.	Comp.	Levenshtein Dist.		Levenshtein Sim.		Avg. score diff.	
		μ	σ	μ	σ	μ	σ
A	R1-R2	55.880	8.019	0.441	0.080	0.009	0.005
A	R1-R3	74.940	5.438	0.251	0.054	0.020	0.011
A	R1-R4	83.100	3.754	0.169	0.038	0.031	0.015
B	R1-R2	54.740	8.024	0.453	0.080	0.008	0.005
B	R1-R3	75.000	5.215	0.250	0.052	0.019	0.011
B	R1-R4	83.680	3.813	0.163	0.038	0.031	0.016
C	R1-R2	55.520	6.044	0.445	0.060	0.007	0.004
C	R1-R3	74.020	4.705	0.260	0.047	0.017	0.008
C	R1-R4	82.720	2.940	0.173	0.029	0.027	0.010

0.1; G3, for $\Theta = 0.2$; G4, for $\Theta = 0.3$. The pruning procedure removes all edges with a weight, in absolute value, lower than the reference threshold, unless the removal of an edge would generate one of the following two anomalies: (i) transition of a node status from connected-born to free-born (see Section 4.2.2), as this would automatically set its acceptability score to 0, significantly altering results; (ii) disconnection of a node from the graph.

In what follows, R1, R2, R3, R4 represent the different rankings calculated on the graphs G1, G2, G3, G4, respectively. The pairs R1-R2, R1-R3, R1-R4 are compared to identify possible savings of computational resources as well as changes in the ranking due to the pruning procedures. Ranking outputs have been encoded into strings built on an alphabet of N symbols. The following metrics have been adopted, where A and B are strings obtained from two rankings under comparison:

- *Levenshtein Distance* $D_L(A, B)$: minimum number of changes (deletions, replacements or insertions) to transform string A into string B;
- *Levenshtein Similarity*, $S_L(A, B) = 1 - \frac{D_L(A,B)}{||A||}$;
- *Average score difference*: average of the numerical difference in absolute value of the score for each argument in the two compared rankings.

Table 4.7 shows the mean and standard deviation values for 50 runs for each configuration A, B, C, where a starting graph G1 is randomly generated and then pruned. The three above-mentioned metrics have been computed

Table 4.8: Performance metrics for the first 5 nodes of the ranking

Config.	**Comp.**	**Levenshtein Dist.**		**Levenshtein Sim.**		**Avg. score diff.**	
		μ	σ	μ	σ	μ	σ
A	R1-R2	2.260	1.572	0.548	0.314	0.009	0.011
A	R1-R3	3.120	1.409	0.376	0.282	0.020	0.018
A	R1-R4	3.680	1.085	0.264	0.217	0.033	0.028
B	R1-R2	1.920	1.495	0.616	0.299	0.008	0.007
B	R1-R3	3.140	1.400	0.372	0.280	0.021	0.024
B	R1-R4	3.780	1.285	0.244	0.257	0.033	0.032
C	R1-R2	2.540	1.459	0.492	0.292	0.007	0.009
C	R1-R3	3.420	1.401	0.316	0.280	0.014	0.014
C	R1-R4	3.880	1.143	0.224	0.229	0.022	0.019

Table 4.9: Performance metrics for the last 5 nodes of the ranking

Config.	**Comp.**	**Levenshtein Dist.**		**Levenshtein Sim.**		**Avg. score diff.**	
		μ	σ	μ	σ	μ	σ
A	R1-R2	2.300	1.500	0.540	0.300	0.008	0.008
A	R1-R3	3.140	1.510	0.372	0.302	0.017	0.013
A	R1-R4	3.540	1.117	0.292	0.223	0.031	0.025
B	R1-R2	2.600	1.549	0.480	0.310	0.011	0.014
B	R1-R3	3.380	1.215	0.324	0.243	0.023	0.021
B	R1-R4	3.920	1.197	0.216	0.239	0.036	0.028
C	R1-R2	2.420	1.511	0.516	0.302	0.005	0.005
C	R1-R3	3.580	1.511	0.516	0.302	0.014	0.011
C	R1-R4	4.100	1.082	0.180	0.216	0.022	0.013

Table 4.10: First and last node comparison

Config.	**Comparison**	**% Same First Node**	**% Same Last Node**
A	R1-R2	74%	68%
A	R1-R3	56%	60%
A	R1-R4	46%	52%
B	R1-R2	82%	68%
B	R1-R3	58%	54%
B	R1-R4	40%	40%
C	R1-R2	70%	74%
C	R1-R3	50%	42%
C	R1-R4	40%	30%

by comparing the ranking of the original graph w.r.t. the one calculated on G2, G3, and G4. Results show a similar trend in the three configurations where the Levenshtein distance increases with increasing threshold value. At the same time, the difference between final scores associated with the same argument is rather small, suggesting the feasibility of pruning.

In pervasive computing agents, decision support algorithms are strongly conditioned by constraints related to time, memory and accuracy. In many cases, top-k query results are what applications are interested in [76] [128]. A careful examination of the most (respectively, least) acceptable arguments in a scenario can be a convenient strategy to meet the typical constraints of pervasive computing. Therefore, the proposed performance metrics have also been evaluated w.r.t. the top and bottom five nodes of the ranking in each configuration. Table 4.8 and 4.9 present the results obtained for the first and last five nodes. On average, 2.26 changes are required to transform the top five nodes of the ranking R1 into the corresponding top five of R2. Also in this case, results of the three configurations show a common trend, essentially similar to the evaluation of the entire graph, with a slight growth of the Levenshtein distance for higher values of Θ.

In addition, Table 4.10 shows the percentage of times the first and last node are the same in the rankings comparing 50 random configurations for each type (A, B, C). For an increasing number of cycles in the graph, result

stability decreases, but in all R1-R3 comparisons (*i.e.*, with pruning about 20% of edges, since weight distribution probability is uniform), the first and last node persist in at least half of the total number of executions.

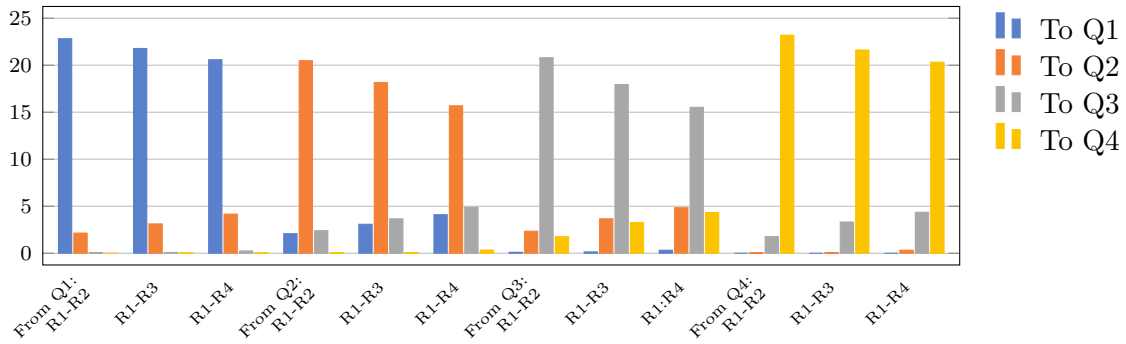
In order to provide a more complete picture, further analysis has been performed on the ranking of all the nodes. First, after dividing the rankings of 100 nodes sorted in descending order with respect to the associated acceptability score into four equal parts (of 25 nodes each), results are reported on the number of nodes of the first quartile Q1 (and then, second Q2, third Q3 and fourth Q4) of the first ranking which are found respectively in the first, second, third and fourth quartiles of the second ranking. This evaluation is repeated for each combination (R1-R2, R1-R3, R1-R4) of ranking pairs in the three graph pruning levels. Table 4.11 shows the average values in percentage with respect to the 50 randomly generated graphs for each configuration. For example, looking at the first row, in the comparison R1-R2 of configuration A, considering the first quartile of R1, on average 91.28% of arguments persist in Q1 of R2, 8.56% move to Q2, only 0.16% to Q3 and no argument in Q1 of R1 has been found in Q4 of R2. Ideally, the higher are the average percentages of Q1-to-Q1, Q2-to-Q2, Q3-to-Q3 and Q4-to-Q4, the lower the impact of pruning is in the ranking results.

Figure 4.7 summarizes the results by means of three histograms showing the distribution of nodes shifting from each quartile of the first ranking to the four quartiles of the second, for each combination of the three configurations. The results appear quite satisfactory. For each quartile of R1, the average number of shifts to the adjacent quartile drops by an order of magnitude; furthermore, there is no significant difference between the three configurations A, B, C. It can further be noted the mean values of Q2-to-Q2 and Q3-to-Q3 (orange and gray bars in Figure 4.7) are generally lower than Q1-to-Q1 and Q4-to-Q4 (blue and yellow bars). For the aforementioned considerations, the higher stability of the top and bottom quartiles is an evidence in favor of exploiting pruning on resource-constrained devices.

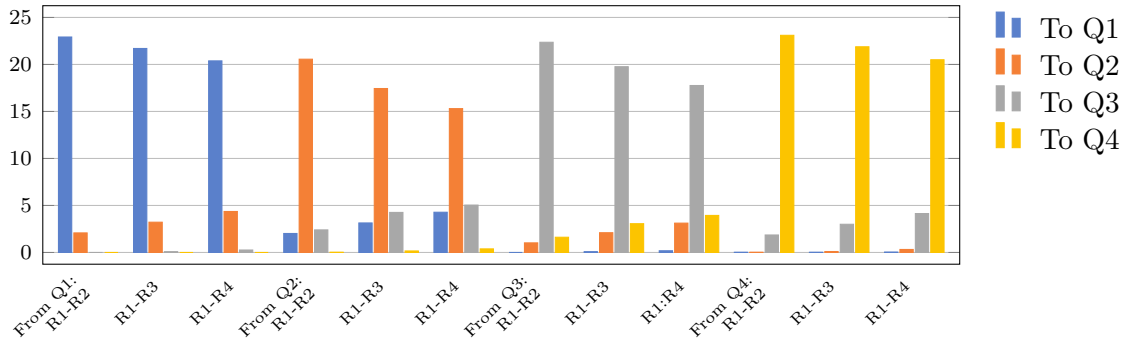
For each graph pair under comparison, a further test has evaluated the number of nodes which shift 0, 1, 2, 3, 4, and 5 positions (either up or down) in the second ranking w.r.t. the first one. Table 4.12 reports the mean and standard deviation values w.r.t. the 50 randomly generated graphs for each

Table 4.11: Quartile shifts in argument rankings due to pruning

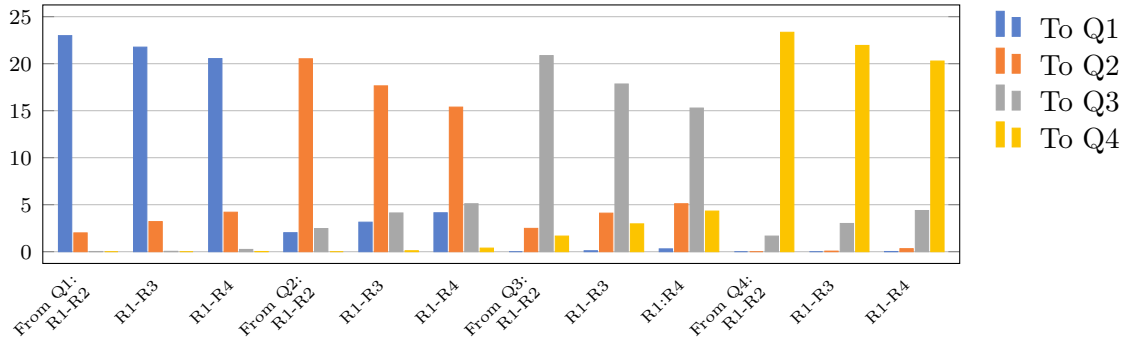
Conf.	Combination	From	To Q1	To Q2	To Q3	To Q4
A	R1-R2	Q1	91,28%	8,56%	0,16%	0
A	R1-R3	Q1	87,12%	12,48%	0,32%	0,08%
A	R1-R4	Q1	82,32%	16,64%	0,96%	0,08%
A	R1-R2	Q2	8,32%	81,92%	9,6%	0,16%
A	R1-R3	Q2	12,32%	72,64%	14,64%	0,24%
A	R1-R4	Q2	16,4%	62,72%	19,52%	1,36%
A	R1-R2	Q3	0,4%	9,36%	83,2%	7,04%
A	R1-R3	Q3	0,56%	14,64%	71,76%	13,04%
A	R1-R4	Q3	1,28%	19,36%	62,08%	17,28%
A	R1-R2	Q4	0	0,16%	7,04%	92,8%
A	R1-R3	Q4	0	0,24%	13,28%	86,48%
A	R1-R4	Q4	0	1,28%	17,44%	81,28%
B	R1-R2	Q1	91,68%	8,32%	0	0
B	R1-R3	Q1	86,8%	12,88%	0,32%	0
B	R1-R4	Q1	81,52%	17,44%	1,04%	0
B	R1-R2	Q2	8,08%	82,24%	9,6%	0,08%
B	R1-R3	Q2	12,56%	69,76%	17,04%	0,64%
B	R1-R4	Q2	17,12%	61,2%	20,16%	1,52%
B	R1-R2	Q3	0	4,08%	89,44%	6,48%
B	R1-R3	Q3	0,32%	8,4%	79,04%	12,24%
B	R1-R4	Q3	0,72%	12,48%	71,04%	15,76%
B	R1-R2	Q4	0,08%	0,08%	7,44%	92,4%
B	R1-R3	Q4	0,08%	0,4%	12%	87,52%
B	R1-R4	Q4	0,16%	1,28%	16,56%	82%
C	R1-R2	Q1	91,92%	8%	0,08%	0
C	R1-R3	Q1	87,04%	12,8%	0,16%	0
C	R1-R4	Q1	82,16%	16,8%	0,96%	0,08%
C	R1-R2	Q2	8,08%	82,08%	9,84%	0
C	R1-R3	Q2	12,56%	70,56%	16,48%	0,4%
C	R1-R4	Q2	16,56%	61,52%	20,4%	1,52%
C	R1-R2	Q3	0	9,92%	83,44%	6,64%
C	R1-R3	Q3	0,4%	16,4%	71,36%	11,84%
C	R1-R4	Q3	1,2%	20,4%	61,12%	17,28%
C	R1-R2	Q4	0	0	6,64%	93,36%
C	R1-R3	Q4	0	0,24%	12%	87,76%
C	R1-R4	Q4	0,08%	1,28%	17,52%	81,12%



(a) Configuration A



(b) Configuration B



(c) Configuration C

Figure 4.7: Argument quartile change in acceptability rankings due to graph pruning

Table 4.12: Argument position shifts due to pruning

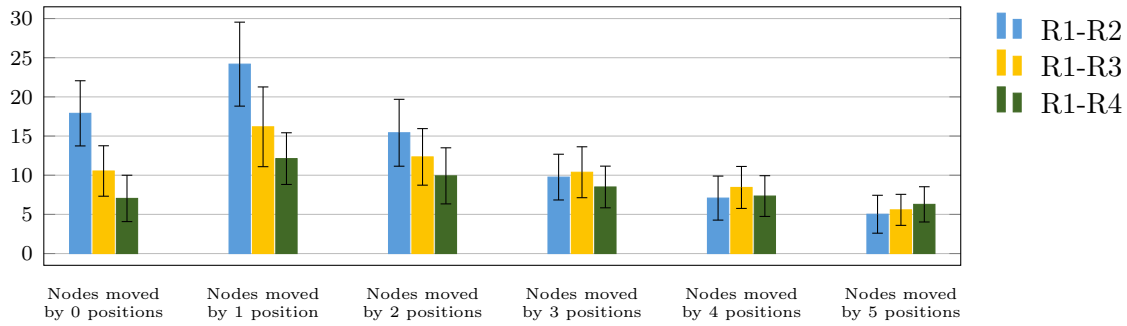
Config.	Comp.	0 positions		1 positions		2 positions		3 positions		4 positions		5 positions	
		μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
A	R1-R2	17.90	4.16	24.18	5.36	15.42	4.27	9.76	2.92	7.08	2.81	5.02	2.42
A	R1-R3	10.54	3.22	16.18	5.09	12.34	3.61	10.38	3.25	8.44	2.68	5.58	1.98
A	R1-R4	7.04	2.96	12.12	3.30	9.92	3.58	8.50	2.66	7.34	2.60	6.28	2.25
B	R1-R2	18.08	4.25	24.34	4.98	15.0	3.31	9.48	2.67	7.18	2.27	4.44	2.51
B	R1-R3	10.64	3.42	15.18	3.91	13.10	3.36	9.42	2.99	7.74	2.76	5.88	2.39
B	R1-R4	7.72	2.91	11.42	3.58	10.12	3.60	8.38	3.41	6.60	2.62	6.36	2.74
C	R1-R2	18.96	5.01	23.98	5.01	16.24	4.17	9.78	3.22	6.40	2.93	5.06	2.22
C	R1-R3	10.24	3.82	15.82	4.14	13.0	3.83	10.08	2.96	7.80	2.57	6.52	2.39
C	R1-R4	7.50	2.39	11.42	3.93	10.16	3.21	8.64	3.29	6.80	2.34	6.46	2.48

Table 4.13: Impact of graph pruning on processing time and memory usage peak

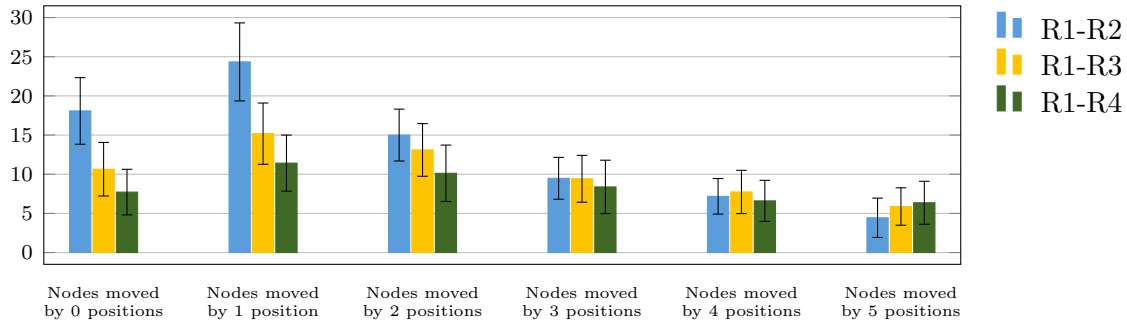
Conf.	Graph	Average # of edges	Execution Time [s]		Memory Usage [MB]	
			μ	Gain	μ	Gain
A	G1	500	152.91	n.a.	147.94	n.a.
A	G2	450	129.19	15.5%	155.64	-5.2%
A	G3	402	115.82	24.3%	138.90	6.1%
A	G4	353	78.30	48.8%	90.42	38.9%
B	G1	500	127.10	n.a.	132.20	n.a.
B	G2	450	102.09	19.7%	114.02	13.8%
B	G3	400	79.82	37.2%	95.54	27.7%
B	G4	350	56.33	55.7%	67.20	49.2%
C	G1	500	70.46	n.a.	43.82	n.a.
C	G2	450	81.48	-15.6%	89.38	-104%
C	G3	400	52.75	25.1%	59.12	-34.9%
C	G4	352	35.01	50.3%	37.50	14.4%

configuration. The three histograms of Figure 4.8 show the average number of nodes that move by a fixed number of positions in the two comparative rankings. For all comparisons (R1-R2, R1-R3, R1-R4) and configurations (A, B, C), the relative majority of arguments shifted by just 1 position; this could be deemed as an encouraging result.

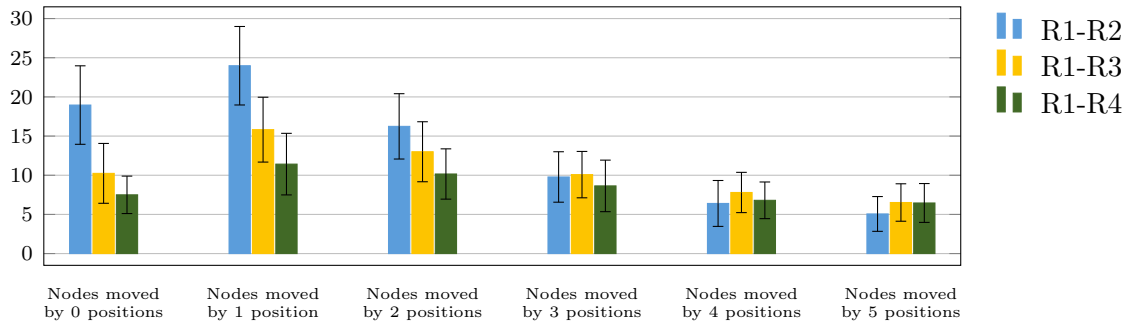
Finally, Table 4.13 reports the mean values μ of ranking semantics processing time and memory usage peak for 50 runs of each pruning threshold in each configuration. In all configurations, the first row (G1) reports the same values presented in Table 4.6. They are compared to those in the next three rows, and the gain is shown. As expected, in these tests pruning with $\Theta = 0.1$ (resp. $0.2, 0.3$) has removed about 10% (resp. $20\%, 30\%$) of edges, since edge weights are random with uniform distribution in the admissible



(a) Configuration A



(b) Configuration B



(c) Configuration C

Figure 4.8: Argument position shifts in acceptability rankings due to graph pruning

interval. The corresponding decrease in execution time is larger, reaching about 50% for $\Theta = 0.3$. Memory peak improvement is more unstable, and particularly graphs G2 for configuration C have execution time and memory consumption which are higher than the unpruned G1.

Overall, the pruning of the BWAF graph edges characterized by weight in absolute value below a certain threshold can be considered as a simple and useful strategy to achieve a trade-off between computational savings in the execution of the ranking with the proposed semantics and limited alteration of the ranking outcomes.

Chapter 5

Conclusions and perspectives

This thesis has analyzed the main critical issues that persist in the Edge Computing paradigm and limit its full potential. The work has proposed innovative tools and effective methodologies to address them.

A novel Cloud-Edge Intelligence distributed framework has been introduced, mainly targeted at IoT-based Cyber-Physical Systems. It employs a microservice architecture and adheres to the Osmotic Computing paradigm to enable opportunistic resource exploitation through the dynamic, flexible deployment of service modules to a variety of devices at the network's edge and/or in the cloud. Modularity is further reinforced by the clear encapsulation of logical components with well defined roles and responsibilities. This allows for a direct mapping with open source software components and COTS devices, increasing feasibility and lowering development costs and time to market. In the proposed approach, model training and prediction tasks can be performed in edge or cloud nodes, but also through cloud-edge collaboration. While cloud devices regularly train larger and more accurate models, delivering them back to the edge, less sophisticated models are trained and utilized at the edge for early response, bandwidth usage reduction and data privacy protection. Experiments on a public domain industrial dataset using a complete prototype platform implementation have been carried out to support the key claims.

Possible improvements include: (i) KRR-based orchestration to dynamically discover the optimal deployment configuration via context-aware se-

mantic matchmaking between ontology-based annotations of microservices and dynamic device descriptions; (ii) exploration of more sophisticated IoT-focused AI algorithms, by enhancing machine learning with semantic technologies [109] and computational argumentation; (iii) integration of the platform prototype with real sensors and actuators in a manufacturing setting or additional challenging IoT-based CPS scenarios, such as (tele)-healthcare, environmental monitoring and urban air mobility.

In order to cater to these challenging scenarios, the thesis has presented a novel structured argumentation framework leveraging Semantic Web languages and Description Logics reasoning. The proposal is based on Bipolar Weighted AF, where arguments are modeled as DL concept expressions and their pairwise relations are automatically evaluated by means of a semantic matchmaking process exploiting non-standard, non-monotonic inference services. The explanation capabilities of the adopted inferences facilitates interpretability of results. Argument acceptability is evaluated by means of a novel propagation-based gradual ranking semantics, which supports cycles in the argumentation graph, satisfies the fading property and complies with ten of the eleven group properties in [18] for well-defined gradual argumentation semantics.

While general-purpose, the proposal aims to be feasible even for pervasive computing agents running on resource-constrained devices. For this reason, convergence and stopping conditions have been integrated in the ranking semantics evaluation algorithm, while the Tiny-ME optimized reasoning engine for mobile and embedded platforms has been leveraged for semantic matchmaking. Moreover, a simplification of the argumentation graph via pruning has been proposed and assessed in experimental evaluations as a possible parametric trade-off between accuracy of ranking semantics outcomes and computational resource usage. A validation of the approach has been carried out by means of a prototypical implementation integrated with the StarCraft II real-time strategy game engine for autonomous decision-making in tactical agents, as RTS game environments simulate many of the challenges of complex and dynamic real-world contexts.

The proposal could be improved in several directions. In the current version, the ranking semantics algorithm is executed in a centralized way, after

building the argumentative graph; ideally, a network of cooperating agents should be able to solve the problem in a coordinated and collaborative way, in order to make larger graphs manageable. Moreover, approaches to avoid a recalculation from scratch of the ranking semantics upon each graph change should be investigated, since information is highly volatile in pervasive contexts. Future work will also aim to attach semantic explanations of relations to the argumentation graph to further improve the interpretability of the framework. Additional research investigations could be conducted to optimize the proposed approach into a predictive AI framework that goes beyond the classic black-box ML approaches and allows enhanced transparency and reliability of the final outcomes, fully exploiting the peculiarities of argumentation for eXplainable AI (XAI) [42]. Finally, experimental campaign in networks of mobile and embedded devices will be required to evaluate the efficiency and effectiveness of the proposal in challenging environments.

Bibliography

[1] Mohammad Aazam and Eui-Nam Huh. Fog computing micro datacenter based dynamic resource estimation and pricing model for iot. In *2015 IEEE 29th International Conference on Advanced Information Networking and Applications*, pages 687–694. IEEE, 2015.

[2] Nasir Abbas, Yan Zhang, Amir Taherkordi, and Tor Skeie. Mobile Edge Computing: A Survey. *IEEE Internet of Things Journal*, 5(1):450–465, 2017.

[3] Alaa Awad Abdellatif, Amr Mohamed, Carla Fabiana Chiasserini, Mounira Tlili, and Aiman Erbad. Edge computing for smart health: Context-aware approaches, opportunities, and challenges. *IEEE Network*, 33(3):196–203, 2019.

[4] Alhanoof Althnian, Duaa AlSaeed, Heyam Al-Baity, Amani Samha, Alanoud Bin Dris, Najla Alzakari, Afnan Abou Elwafa, and Heba Kurdi. Impact of Dataset Size on Classification Performance: An Empirical Evaluation in the Medical Domain. *Applied Sciences*, 11(2):796, 2021.

[5] Abdulmalik Alwarafy, Khaled A. Al-Thelaya, Mohamed Abdallah, Jens Schneider, and Mounir Hamdi. A survey on security and privacy issues in edge-computing-assisted Internet of Things. *IEEE Internet of Things Journal*, 8(6):4004–4022, 2020.

[6] Amira A. Amer, Ihab E. Talkhan, Reem Ahmed, and Tawfik Ismail. An optimized collaborative scheduling algorithm for prioritized tasks

with shared resources in mobile-edge and cloud computing systems. *Mobile Networks and Applications*, 27(4):1444–1460, 2022.

[7] Leila Amgoud and Jonathan Ben-Naim. Ranking-based semantics for argumentation frameworks. In *International Conference on Scalable Uncertainty Management*, pages 134–147. Springer, 2013.

[8] Leila Amgoud and Jonathan Ben-Naim. Evaluation of arguments in weighted bipolar graphs. *International Journal of Approximate Reasoning*, 99:39–55, 2018.

[9] Leila Amgoud, Jonathan Ben-Naim, Dragan Doder, and Srdjan Vesic. Ranking arguments with compensation-based semantics. In *Fifteenth International Conference on the Principles of Knowledge Representation and Reasoning*, pages 12–21, 2016.

[10] Leila Amgoud, Claudette Cayrol, Marie-Christine Lagasquie-Schiex, and Pierre Livet. On bipolarity in argumentation frameworks. *International Journal of Intelligent Systems*, 23(10):1062–1093, 2008.

[11] Leila Amgoud, Dragan Doder, and Srdjan Vesic. Evaluation of argument strength in attack graphs: Foundations and semantics. *Artificial Intelligence*, 302:103607, 2022.

[12] Hany F. Atlam, Robert J. Walters, and Gary B. Wills. Fog computing and the Internet of Things: A review. *big data and cognitive computing*, 2(2):10, 2018.

[13] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider. *The Description Logic Handbook*. Cambridge University Press, 2002.

[14] Andrew Banks, Ed Briggs, Ken Borgendale, and Rauhl Gupta. MQTT Version 5.0. Technical report, OASIS: Burlington, MA, USA, March 2019.

[15] Pietro Baroni, Paul E. Dunne, and Massimiliano Giacomin. On the resolution-based family of abstract argumentation semantics and its grounded instance. *Artificial Intelligence*, 175(3-4):791–813, 2011.

[16] Pietro Baroni and Massimiliano Giacomin. Solving semantic problems with odd-length cycles in argumentation. In *European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, pages 440–451. Springer, 2003.

[17] Pietro Baroni and Massimiliano Giacomin. On principle-based evaluation of extension-based argumentation semantics. *Artificial Intelligence*, 171(10):675–700, 2007.

[18] Pietro Baroni, Antonio Rago, and Francesca Toni. How many properties do we need for gradual argumentation? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[19] Trevor J. M. Bench-Capon and Paul E. Dunne. Argumentation in artificial intelligence. *Artificial Intelligence*, 171(10-15):619–641, 2007.

[20] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic Web. *Scientific American*, 284(5):28–37, 2001.

[21] Philippe Besnard, Alejandro Garcia, Anthony Hunter, Sanjay Modgil, Henry Prakken, Guillermo Simari, and Francesca Toni. Introduction to structured argumentation. *Argument & Computation*, 5(1):1–4, 2014.

[22] Philippe Besnard and Anthony Hunter. A logic-based theory of deductive arguments. *Artificial Intelligence*, 128(1-2):203–235, 2001.

[23] Philippe Besnard and Anthony Hunter. *Elements of Argumentation*. MIT Press, 2008.

[24] Chatschik Bisdikian. An overview of the Bluetooth wireless technology. *IEEE Communications Magazine*, 39(12):86–94, 2001.

[25] James Bond. *The enterprise cloud: Best practices for transforming legacy IT*. O'Reilly Media, Inc., 2015.

[26] Andrei Bondarenko, Phan Minh Dung, Robert A. Kowalski, and Francesca Toni. An abstract, argumentation-theoretic approach to default reasoning. *Artificial Intelligence*, 93(1-2):63–101, 1997.

[27] Elise Bonzon, Jérôme Delobelle, Sébastien Konieczny, and Nicolas Maudet. Argumentation ranking semantics based on propagation. *6th International Conference on Computational Models of Argument (COMMA)*, 16:139–150, 2016.

[28] Elise Bonzon, Jérôme Delobelle, Sébastien Konieczny, and Nicolas Maudet. A comparative study of ranking-based semantics for abstract argumentation. In *Thirtieth AAAI Conference on Artificial Intelligence*, pages 914–920, 2016.

[29] Elise Bonzon, Jérôme Delobelle, Sébastien Konieczny, and Nicolas Maudet. A parametrized ranking-based semantics compatible with persuasion principles. *Argument & Computation*, 12(1):49–85, 2021.

[30] Alexander Borgida. Description logics in data management. *IEEE Transactions on Knowledge and Data Engineering*, 7(5):671–682, 1995.

[31] Amel Bouzeghoub, Said Jabbour, Yue Ma, and Badran Raddaoui. Handling conflicts in uncertain ontologies using deductive argumentation. In *International Conference on Web Intelligence*, pages 65–72. ACM, 2017.

[32] Noam Brown and Tuomas Sandholm. Superhuman AI for multiplayer poker. *Science*, 365(6456):885–890, 2019.

[33] Martin Caminada. Semi-stable semantics. *COMMA*, 144:121–130, 2006.

[34] Martin Caminada. Comparing two unique extension semantics for formal argumentation: ideal and eager. In *Proceedings of the 19th Belgian-Dutch conference on artificial intelligence (BNAIC 2007)*, pages 81–87. Utrecht University Press, 2007.

[35] Lorenzo Carnevale, Antonio Celesti, Antonino Galletta, Schahram Dustdar, and Massimo Villari. Osmotic computing as a distributed multi-agent system: The Body Area Network scenario. *Internet of Things*, 5:130–139, 2019.

[36] Claudette Cayrol and Marie-Christine Lagasquie-Schiex. On the acceptability of arguments in bipolar argumentation frameworks. In *European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty, ECSQARU*, volume 3571 of *Lecture Notes in Computer Science*, pages 378–389. Springer, 2005.

[37] Tanusree Chatterjee, Sushmita Ruj, and Sipra Das Bit. Security issues in named data networks. *Computer*, 51(1):66–75, 2018.

[38] Simona Colucci, Tommaso Di Noia, Agnese Pinto, Azzurra Ragone, Michele Ruta, and Eufemia Tinelli. A Non-Monotonic Approach to Semantic Matchmaking and Request Refinement in E-Marketplaces. *International Journal of Electronic Commerce*, 12(2):127–154, 2007.

[39] Marco Conti and Mohan Kumar. Opportunities in opportunistic computing. *Computer*, 43(01):42–50, 2010.

[40] Sylvie Coste-Marquis, Caroline Devred, and Pierre Marquis. Prudent semantics for argumentation frameworks. In *17th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'05)*, pages 5–pp. IEEE, 2005.

[41] Sylvie Coste-Marquis, Caroline Devred, and Pierre Marquis. Symmetric argumentation frameworks. In *European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, pages 317–328. Springer, 2005.

[42] Kristijonas Čyras, Antonio Rago, Emanuele Albini, Pietro Baroni, and Francesca Toni. Argumentative xai: a survey. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 4392–4399. International Joint Conferences on Artificial Intelligence Organization, 2021.

[43] Paul Delory, Lydia Leong, Tony Iams, Matthew Brisse, Douglas Toombs, Angelina Troy, Stanton Cole, Fintan Quinn, Mohini Dukes, and Marco Meinardi. 2022 planning guide for cloud and edge computing. Technical report, Gartner, October 2021.

[44] Shuiguang Deng, Hailiang Zhao, Weijia Fang, Jianwei Yin, Schahram Dustdar, and Albert Y Zomaya. Edge Intelligence: the Confluence of Edge Computing and Artificial Intelligence. *IEEE Internet of Things Journal*, 7(8):7457–7469, 2020.

[45] Peter Deutsch and Jean-Loup Gailly. RFC1950: ZLIB Compressed Data Format Specification Version 3.3. Technical report, Internet Engineering Task Force, USA, 1996.

[46] Tommaso Di Noia, Eugenio Di Sciascio, and Francesco M. Donini. Semantic matchmaking as non-monotonic reasoning: A description logic approach. *Journal of Artificial Intelligence Research (JAIR)*, 29:269–307, 2007.

[47] Tommaso Di Noia, Eugenio Di Sciascio, Francesco M. Donini, and Marina Mongiello. A System for Principled Matchmaking in an Electronic Marketplace. *International Journal of Electronic Commerce*, 8(4):9–37, 2004.

[48] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. Reasoning in description logics. *Principles of Knowledge representation*, 1:191–236, 1996.

[49] Phan Minh Dung. On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and N-Person Games. *Artificial intelligence*, 77(2):321–357, 1995.

[50] Phan Minh Dung, Paolo Mancarella, and Francesca Toni. A dialectic procedure for sceptical, assumption-based argumentation. *Comma*, 144:145–156, 2006.

[51] Paul E. Dunne, Anthony Hunter, Peter McBurney, Simon Parsons, and Michael J. Wooldridge. Inconsistency tolerance in weighted argument systems. In *AAMAS (2)*, pages 851–858, 2009.

[52] Paul E. Dunne, Anthony Hunter, Peter McBurney, Simon Parsons, and Michael J. Wooldridge. Weighted argument systems: Basic definitions,

algorithms, and complexity results. *Artificial Intelligence*, 175(2):457 – 486, 2011.

[53] Wolfgang Dvořák and Sarah Alice Gaggl. Computational aspects of cf2 and stage2 argumentation semantics. In *Computational Models of Argument*, pages 273–284. IOS Press, 2012.

[54] Sinem Coleri Ergen. Zigbee/ieee 802.15. 4 summary. *UC Berkeley, September*, 10(17):11, 2004.

[55] Timofey Ermilov, Ali Khalili, and Sören Auer. Ubiquitous semantic applications: a systematic literature review. *International Journal on Semantic Web and Information Systems*, 10(1):66–99, 2014.

[56] Corrado Fasciano, Michele Ruta, and Floriano Scioscia. Semantic matchmaking for argumentative intelligence in ubiquitous computing. In *Current Trends in Web Engineering*, pages 137–148. Springer Nature Switzerland, 2023.

[57] Alejandro J. García and Guillermo R. Simari. Defeasible logic programming: An argumentative approach. *Theory and Practice of Logic Programming*, 4(1-2):95–138, 2004.

[58] Salvador García, Julián Luengo, and Francisco Herrera. *Data preprocessing in data mining*. Springer, 2015.

[59] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT press Cambridge, 2016.

[60] Dominik Grzelak, Johannes Mey, and Uwe Aßmann. Design and Concept of an Osmotic Analytics Platform based on R Container. In *Proceedings of the International Conference on Foundations of Computer Science*, pages 29–35. UCMSS, 2018.

[61] Nancy Gulati and Pankaj Deep Kaur. A game theoretic approach for conflict resolution in argumentation enabled social IoT networks. *Ad Hoc Networks*, 107:102222, 2020.

[62] Nancy Gulati and Pankaj Deep Kaur. An argumentation enabled decision making approach for Fall Activity Recognition in Social IoT based Ambient Assisted Living systems. *Future Generation Computer Systems*, 122:82–97, 2021.

[63] Guus Schreiber and Yves Raimond. RDF 1.1 Primer. W3C Recommendation, W3C, June 2014. `https://www.w3.org/TR/rdf11-primer/`.

[64] Volker Haarslev and Ralf Möller. Racer system description. *Automated Reasoning*, pages 701–705, 2001.

[65] Stella Heras, Vicente Botti, and Vicente Julián. An Ontological-based Knowledge-representation Formalism for Case-based Argumentation. *Information Systems Frontiers*, 17(4):779–798, August 2015.

[66] Matthew Horridge and Peter Patel-Schneider. OWL 2 Web Ontology Language Manchester Syntax (Second Edition). W3C note, W3C, December 2012. `http://www.w3.org/TR/owl2-manchester-syntax/`.

[67] Congfeng Jiang, Tiantian Fan, Honghao Gao, Weisong Shi, Liangkai Liu, Christophe Cérin, and Jian Wan. Energy aware edge computing: A survey. *Computer Communications*, 151:556–580, 2020.

[68] Kuljeet Kaur, Sahil Garg, Georges Kaddoum, Syed Hassan Ahmed, and Dushantha Nalin K. Jayakody. En-OsCo: Energy-aware Osmotic Computing Framework using Hyper-heuristics. In *Proceedings of the ACM MobiHoc Workshop on Pervasive Systems in the IoT Era*, pages 19–24, 2019.

[69] Yevgeny Kazakov. Consequence-driven reasoning for Horn $\mathcal{SHIQ}$ ontologies. In *Twenty-First International Joint Conference on Artificial Intelligence*, 2009.

[70] Yevgeny Kazakov and Pavel Klinov. Experimenting with ELK Reasoner on Android. In *Proc. of 2nd International Workshop on OWL Reasoner Evaluation (ORE'13), Ulm (Germany)*, pages 68–74, 2013.

[71] Wazir Zada Khan, Ejaz Ahmed, Saqib Hakak, Ibrar Yaqoob, and Arif Ahmed. Edge computing: A survey. *Future Generation Computer Systems*, 97:219–235, 2019.

[72] Taehun Kim, Insuk Park, Soon J. Hyun, and Dongman Lee. MiRE4OWL: Mobile Rule Engine for OWL. In *Computer Software and Applications Conf. Workshops (COMPSACW), 2010 IEEE 34th Annual*, pages 317–322. IEEE, 2010.

[73] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[74] Thomas Kleemann and Alex Sinner. User Profiles and Matchmaking on Mobile Phones. In Oscar Bartenstein, editor, *Proc. of 16th Int. Conf. on Applications of Declarative Programming and Knowledge Management INAP2005, Fukuoka*, 2005.

[75] Fernando Koch. 3APL-M platform for deliberative agents in mobile devices. In *Proceedings of Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, page 154. ACM, 2005.

[76] Kostas Kolomvatsos and Christos Anagnostopoulos. A Proactive Statistical Model Supporting Services and Tasks Management in Pervasive Applications. *IEEE Transactions on Network and Service Management*, 2022.

[77] Lei Li and Ian Horrocks. A software framework for matchmaking based on semantic web technology. *International Journal of Electronic Commerce*, 8(4):39–60, 2004.

[78] Shancang Li, Li Da Xu, and Shanshan Zhao. The Internet of Things: a survey. *Information Systems Frontiers*, 17(2):243–259, 2015.

[79] Wei Li, Xinghui You, Yingying Jiang, Jun Yang, and Long Hu. Opportunistic computing offloading in edge clouds. *Journal of Parallel and Distributed Computing*, 123:69–76, 2019.

[80] Xinghua Li, Ting Chen, Qingfeng Cheng, Siqi Ma, and Jianfeng Ma. Smart applications in edge computing: Overview on authentication and data security. *IEEE Internet of Things Journal*, 8(6):4063–4080, 2020.

[81] Yuan-Fang Li, Jeff Z. Pan, Manfred Hauswirth, and Hai Nguyen. The Ubiquitous Semantic Web: Promises, Progress and Challenges. In *Web-Based Services: Concepts, Methodologies, Tools, and Applications*, pages 272–289. IGI Global, Hershey, PA, USA, 2016.

[82] Yujin Li and Wenye Wang. The unheralded power of cloudlet computing in the vicinity of mobile devices. In *2013 IEEE Global Communications Conference (GLOBECOM)*, pages 4994–4999. IEEE, 2013.

[83] Marco Lippi, Marco Mamei, Stefano Mariani, and Franco Zambonelli. An argumentation-based perspective over the social IoT. *IEEE Internet of Things Journal*, 5(4):2537–2547, 2018.

[84] Antonella Longo, Andrea De Matteis, and Marco Zappatore. Urban pollution monitoring based on Mobile Crowd Sensing: an osmotic computing approach. In *2018 IEEE 4th International Conference on Collaboration and Internet Computing*, pages 380–387. IEEE, 2018.

[85] Giuseppe Loseto, Floriano Scioscia, Michele Ruta, Filippo Gramegna, Saverio Ieva, Corrado Fasciano, Ivano Bilenchi, and Davide Loconte. Osmotic Cloud-Edge Intelligence for IoT-Based Cyber-Physical Systems. *Sensors*, 22(6):2166, 2022.

[86] Ellie Lovellette, Henry Hexmoor, and Kane Rodriguez. Automated argumentation for collaboration among cyber-physical system actors at the edge of the Internet of Things. *Internet of Things*, 5:84–96, 2019.

[87] Eduardo Magalhães Oliveira. Quality Prediction in a Mining Process. https://www.kaggle.com/edumagalhaes/quality-prediction-in-a-mining-process. Last accessed: 2022-12-02.

[88] Javier Mendez, Kay Bierzynski, M. P. Cuéllar, and Diego P. Morales. Edge intelligence: Concepts, architectures, applications and future directions. *ACM Transactions on Embedded Computing Systems (TECS)*, 2022.

[89] Antti P. Miettinen and Jukka K. Nurminen. Energy efficiency of mobile clients in cloud computing. *HotCloud*, 10(4-4):19, 2010.

[90] Martín O. Moguillansky and Guillermo R. Simari. A Generalized Abstract Argumentation Framework for Inconsistency-tolerant Ontology Reasoning. *Expert Systems with Applications*, 64(C):141–168, December 2016.

[91] Ahsan Morshed, Prem Prakash Jayaraman, Timos Sellis, Dimitrios Georgakopoulos, Massimo Villari, and Rajiv Ranjan. Deep Osmosis: Holistic Distributed Deep Learning in Osmotic Computing. *IEEE Cloud Computing*, 4(6):22–32, 2017.

[92] Boris Motik, Ian Horrocks, and Su Myeon Kim. Delta-Reasoner: a Semantic Web Reasoner for an Intelligent Mobile Platform. In *Twenty-first International World Wide Web Conference (WWW 2012)*, pages 63–72. ACM, 2012.

[93] Nir Oren and Timothy J. Norman. Semantics for evidence-based argumentation. In *Computational Models of Argument: Proceedings of COMMA 2008*, pages 276–284. IOS Press, 2008.

[94] Ercan Oztemel and Samet Gursev. Literature review of Industry 4.0 and related technologies. *Journal of Intelligent Manufacturing*, 31(1):127–182, 2020.

[95] Alberto Pacheco, Pablo Cano, Ever Flores, Edgar Trujillo, and Pedro Marquez. A smart classroom based on deep learning and osmotic iot

computing. In *2018 Congreso Internacional de Innovación y Tendencias en Ingeniería (CONIITI)*, pages 1–5. IEEE, 2018.

[96] Bijan Parsia, Sebastian Rudolph, Markus Krötzsch, Peter Patel-Schneider, and Pascal Hitzler. OWL 2 Web Ontology Language Primer (Second Edition). W3C Recommendation, W3C, December 2012. `http://www.w3.org/TR/owl2-primer/`.

[97] Andrea Pazienza, Stefano Ferilli, and Floriana Esposito. Constructing and evaluating bipolar weighted argumentation frameworks for online debating systems. In *1st Workshop on Advances In Argumentation In Artificial Intelligence, XVI International Conference of the Italian Association for Artificial Intelligence (AI³@AI*IA)*, pages 111–125, 2017.

[98] Laércio Pioli, Carina F. Dorneles, Douglas D. J. de Macedo, and Mario A. R. Dantas. An overview of data reduction solutions at the edge of IoT systems: a systematic mapping of the literature. *Computing*, 104(8):1867–1889, 2022.

[99] Tie Qiu, Jiancheng Chi, Xiaobo Zhou, Zhaolong Ning, Mohammed Atiquzzaman, and Dapeng Oliver Wu. Edge computing in Industrial Internet of Things: Architecture, advances and challenges. *IEEE Communications Surveys & Tutorials*, 22(4):2462–2488, 2020.

[100] Thomas Rausch, Schahram Dustdar, and Rajiv Ranjan. Osmotic Message-Oriented Middleware for the Internet of Things. *IEEE Cloud Computing*, 5(2):17–25, 2018.

[101] Dipankar Raychaudhuri, Kiran Nagaraja, and Arun Venkataramani. Mobilityfirst: a robust and trustworthy mobility-centric architecture for the future Internet. *ACM SIGMOBILE Mobile Computing and Communications Review*, 16(3):2–13, 2012.

[102] Chris Reed, Katarzyna Budzynska, Rory Duthie, Mathilde Janier, Barbara Konat, John Lawrence, Alison Pease, and Mark Snaith. The argument web: An online ecosystem of tools, systems and services for argumentation. *Philosophy & Technology*, 30(2):137–160, 2017.

[103] Yara Rizk, Mariette Awad, and Edward W. Tunstel. Decision making in multiagent systems: A survey. *IEEE Transactions on Cognitive and Developmental Systems*, 10(3):514–529, 2018.

[104] Michele Ruta, Eugenio Di Sciascio, and Floriano Scioscia. Concept abduction and contraction in semantic-based P2P environments. *Web Intelligence and Agent Systems*, 9(3):179–207, 2011.

[105] Michele Ruta, Floriano Scioscia, Ivano Bilenchi, Filippo Gramegna, Giuseppe Loseto, Saverio Ieva, and Agnese Pinto. A multiplatform reasoning engine for the Semantic Web of Everything. *Journal of Web Semantics*, 73:100709, 2022.

[106] Michele Ruta, Floriano Scioscia, Eugenio Di Sciascio, and Ivano Bilenchi. OWL API for iOS: early implementation and results. In *13th OWL: Experiences and Directions Workshop and 5th OWL reasoner evaluation workshop (OWLED - ORE 2016)*, volume 10161 of *Lecture Notes in Computer Science*, pages 141–152, Berlin, Germany, Nov 2016. W3C, Springer.

[107] Michele Ruta, Floriano Scioscia, Filippo Gramegna, Ivano Bilenchi, and Eugenio Di Sciascio. Mini-ME Swift: the first OWL reasoner for iOS. In *16th Extended Semantic Web Conference*, number 11503 in Lecture Notes in Computer Science, pages 298–313. Springer, 2019.

[108] Michele Ruta, Floriano Scioscia, Giuseppe Loseto, Filippo Gramegna, Saverio Ieva, Agnese Pinto, and Eugenio Di Sciascio. Social Internet of Things for domotics: A knowledge-based approach over LDP-CoAP. *Semantic Web*, 9(6):781–802, 2018.

[109] Michele Ruta, Floriano Scioscia, Giuseppe Loseto, Agnese Pinto, and Eugenio Di Sciascio. Machine learning in the Internet of Things: A semantic-enhanced approach. *Semantic Web*, 10(1):183–204, 2019.

[110] Mahadev Satyanarayanan. The emergence of edge computing. *Computer*, 50(1):30–39, 2017.

[111] Guus Schreiber and Fabien Gandon. RDF 1.1 XML syntax. W3C recommendation, W3C, February 2014. `http://www.w3.org/TR/rdf-syntax-grammar/`.

[112] Floriano Scioscia and Michele Ruta. Building a Semantic Web of Things: issues and perspectives in information compression. In *Semantic Web Information Management (SWIM'09). In Proceedings of the 3rd IEEE International Conference on Semantic Computing (ICSC 2009)*, pages 589–594. IEEE Computer Society, 2009.

[113] Floriano Scioscia, Michele Ruta, Giuseppe Loseto, Filippo Gramegna, Saverio Ieva, Agnese Pinto, and Eugenio Di Sciascio. A mobile matchmaker for the Ubiquitous Semantic Web. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 10(4):77–100, 2014.

[114] Floriano Scioscia, Michele Ruta, Giuseppe Loseto, Filippo Gramegna, Saverio Ieva, Agnese Pinto, and Eugenio Di Sciascio. Mini-ME matchmaker and reasoner for the Semantic Web of Things. In *Innovations, Developments, and Applications of Semantic Web and Information Systems*, pages 262–294. IGI Global, Hershey, PA, USA, 2018.

[115] Christian Seitz and René Schönfelder. Rule-based OWL reasoning for specific embedded devices. In *International Semantic Web Conference*, pages 237–252, Berlin, Germany, 2011. Springer.

[116] Dimitrios Serpanos. The Cyber-Physical Systems Revolution. *Computer*, 51(3):70–73, 2018.

[117] Kewei Sha, T. Andrew Yang, Wei Wei, and Sadegh Davari. A survey of edge computing-based designs for IoT security. *Digital Communications and Networks*, 6(2):195–202, 2020.

[118] Vishal Sharma, Dushantha Nalin K. Jayakody, and Marwa Qaraqe. Osmotic computing-based service migration and resource scheduling in Mobile Augmented Reality Networks (MARN). *Future Generation Computer Systems*, 102:723–737, 2020.

[119] Vishal Sharma, Ilsun You, Ravinder Kumar, and Pankoo Kim. Computational offloading for efficient trust management in pervasive online social networks using osmotic computing. *IEEE Access*, 5:5084–5103, 2017.

[120] Robert D. C. Shearer, Boris Motik, and Ian Horrocks. HermiT: A highly-efficient OWL reasoner. In *Proc. of the 5th Int. Workshop on OWL: Experiences and Directions (OWLED 2008)*, pages 26–27, 2008.

[121] Zach Shelby, Klaus Hartke, and Carsten Bormann. The Constrained Application Protocol (CoAP). RFC 7252, IETF: Wilmington, DE, USA, 2014.

[122] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5):637–646, 2016.

[123] Alex Sinner and Thomas Kleemann. KRHyper - In Your Pocket. In *Proc. of 20th Int. Conf. on Automated Deduction (CADE-20)*, pages 452–457, Tallinn, Estonia, July 2005.

[124] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A practical OWL-DL reasoner. *Web Semantics: science, services and agents on the World Wide Web*, 5(2):51–53, 2007.

[125] Steffen Staab and Rudi Studer. *Handbook on ontologies*. Springer Science & Business Media, 2010.

[126] Luke Steller and Shonali Krishnaswamy. Pervasive Service Discovery: mTableaux Mobile Reasoning. In *International Conference on Semantic Systems (I-Semantics). Graz, Austria*, 2008.

[127] Wei Tai, John Keeney, and Declan O'Sullivan. COROR: a composable rule-entailment OWL reasoner for resource-constrained devices. *Rule-Based Reasoning, Programming, and Applications*, pages 212–226, 2011.

[128] Chiu C. Tan, Bo Sheng, Haodong Wang, and Qun Li. Microsearch: A search engine for embedded devices used in pervasive computing. *ACM Transactions on Embedded Computing Systems (TECS)*, 9(4):1–29, 2010.

[129] Christoph Tempich, H. Sofia Pinto, York Sure, and Steffen Staab. An argumentation ontology for DIstributed, Loosely-controlled and evolvInG Engineering processes of oNTologies (DILIGENT). In *European Semantic Web Conference*, pages 241–256. Springer, 2005.

[130] Daniele Tovazzi, Francescomaria Faticanti, Domenico Siracusa, Claudio Peroni, Silvio Cretti, and Tommaso Gazzini. GEM-Analytics: Cloud-to-Edge AI-Powered Energy Management. In *International Conference on the Economics of Grids, Clouds, Systems, and Services*, pages 57–66. Springer, 2020.

[131] Dmitry Tsarkov and Ian Horrocks. FaCT++ description logic reasoner: System description. *Automated Reasoning*, pages 292–297, 2006.

[132] Blesson Varghese, Nan Wang, Sakil Barbhuiya, Peter Kilpatrick, and Dimitrios S. Nikolopoulos. Challenges and opportunities in edge computing. In *2016 IEEE International conference on Smart Cloud*, pages 20–26. IEEE, 2016.

[133] Bart Verheij. Two approaches to dialectical argumentation: admissible sets and argumentation stages. *Proceedings of the Eighth Dutch Conference on Artificial Intelligence (NAIC)*, 96:357–368, 1996.

[134] Massimo Villari, Maria Fazio, Schahram Dustdar, Omer Rana, Devki Nandan Jha, and Rajiv Ranjan. Osmosis: The osmotic computing platform for microelements in the cloud, edge, and Internet of Things. *Computer*, 52(8):14–26, 2019.

[135] Massimo Villari, Maria Fazio, Schahram Dustdar, Omer Rana, and Rajiv Ranjan. Osmotic Computing: A New Paradigm for Edge/Cloud Integration. *IEEE Cloud Computing*, 3(6):76–83, 2016.

[136] Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, et al. Starcraft II: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*, 2017.

[137] Danny Weyns, Andrea Omicini, and James Odell. Environment as a first class abstraction in multiagent systems. *Autonomous Agents and Multi-Agent Systems*, 14(1):5–30, 2007.

[138] William Van Woensel and Syed Sibte Raza Abidi. Optimizing semantic reasoning on memory-constrained platforms using the RETE algorithm. In *European Semantic Web Conference*, pages 682–696. Springer, 2018.

[139] Marilyn Wolf. Machine learning + distributed IoT = Edge Intelligence. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 1715–1719. IEEE, 2019.

[140] Liuwen Yu and Leendert Van der Torre. A principle-based approach to bipolar argumentation. In *18th International Workshop on Non-Monotonic Reasoning (NMR)*, volume 227, 2020.

[141] Wei Yu, Fan Liang, Xiaofei He, William Grant Hatcher, Chao Lu, Jie Lin, and Xinyu Yang. A survey on the edge computing for the Internet of Things. *IEEE Access*, 6:6900–6919, 2017.

[142] Roberto Yus, Carlos Bobed, Guillermo Esteban, Fernando Bobillo, and Eduardo Mena. Android goes semantic: DL reasoners on smartphones. In *Proceedings of 2nd International Workshop on OWL Reasoner Evaluation (ORE'13), Ulm (Germany)*, pages 46–52, 2013.

[143] Ke Zhang, Yuming Mao, Supeng Leng, Quanxin Zhao, Longjiang Li, Xin Peng, Li Pan, Sabita Maharjan, and Yan Zhang. Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks. *IEEE Access*, 4:5896–5907, 2016.

[144] Lixia Zhang, Deborah Estrin, Jeffrey Burke, Van Jacobson, James D. Thornton, Diana K. Smetters, Beichuan Zhang, Gene Tsudik, Dan Massey, Christos Papadopoulos, et al. Named Data Networking (NDN) project. *Technical Report NDN-0001, Xerox Palo Alto Research Center-PARC*, 157:1–158, 2010.

[145] Xingzhou Zhang, Yifan Wang, Sidi Lu, Liangkai Liu, Lanyu Xu, and Weisong Shi. OpenEI: An open framework for edge intelligence. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 1840–1851. IEEE, 2019.

[146] Zhi Zhou, Xu Chen, En Li, Liekang Zeng, Ke Luo, and Junshan Zhang. Edge intelligence: Paving the last mile of artificial intelligence with edge computing. *Proceedings of the IEEE*, 107(8):1738–1762, 2019.

List of publications

Journal Articles

1. Giuseppe Loseto, Floriano Scioscia, Michele Ruta, Filippo Gramegna, Saverio Ieva, Corrado Fasciano, Ivano Bilenchi, and Davide Loconte. *Osmotic Cloud-Edge Intelligence for IoT-Based Cyber-Physical Systems*, In Sensors, MDPI, volume 22 (6), article no. 2166, 2022

2. Andrea Pazienza, Roberto Anglani, Corrado Fasciano, Corrado Tatulli, and Felice Vitulano. *Evolving and explainable clinical risk assessment at the edge*, In Evolving Systems, Springer, volume 13 (3), pp. 403-422, 2022

Peer-Reviewed Conference Papers

1. Corrado Fasciano, Michele Ruta, and Floriano Scioscia. *Semantic Matchmaking for Argumentative Intelligence in Ubiquitous Computing*, In Current Trends in Web Engineering, Springer Nature, pp. 137-148, 2023

2. Corrado Fasciano, Giuseppe Loseto, Floriano Scioscia, and Michele Ruta. *Semantic-enhanced Argumentation Framework for Autonomous Decision-making in Smart Mobility*, In 8th Italian conference on ICT for Smart Cities & Communities (I-CiTies 2022), 2022

3. Carmelo Ardito, Tommaso Di Noia, Corrado Fasciano, Domenico Lofù, Nicola Macchiarulo, Giulio Mallardi, Andrea Pazienza, and Felice Vitulano. *An Edge Ambient Assisted Living Process for Clinical Pathway*, In Ambient Assisted Living, ForItAAL 2020, Springer, volume 884, pp. 363-374, 2022

4. Tommaso Di Noia, Corrado Fasciano, Domenico Lofù, Giulio Mallardi, Graziano Pappadà, Corrado Tatulli, and Felice Vitulano. *INSTAMED: an Integrated Platform for the Advanced Automation of Diagnosis in*

Precision Medicine, In 7th Italian conference on ICT for Smart Cities & Communities (I-CiTies 2021), 2021

5. Carmelo Ardito, Tommaso Di Noia, Corrado Fasciano, Domenico Lofù, Nicola Macchiarulo, Giulio Mallardi, Andrea Pazienza, and Felice Vitulano. *Management at the edge of situation awareness during patient telemonitoring*, In Advances in Artificial Intelligence, AIxIA 2020, Springer, volume 12414, pp. 372-387, 2020

6. Carmelo Ardito, Tommaso Di Noia, Corrado Fasciano, Domenico Lofù, Nicola Macchiarulo, Giulio Mallardi, Andrea Pazienza, and Felice Vitulano. *Towards a Situation Awareness for eHealth in Ageing Society*, In Italian Workshop on Artificial Intelligence for an Ageing Society 2020 co-located with 19th International Conference of the Italian Association for Artificial Intelligence (AIxIA 2020), 2020

7. Andrea Pazienza, Roberto Anglani, Giulio Mallardi, Corrado Fasciano, Pietro Noviello, Corrado Tatulli, and Felice Vitulano. *Adaptive critical care intervention in the internet of medical things*, In 2020 IEEE International Conference on Evolving and Adaptive Intelligent Systems (EAIS 2020), pp. 1-8, 2020

8. Andrea Pazienza, Giulio Mallardi, Corrado Fasciano, and Felice Vitulano. *Artificial Intelligence on Edge Computing: a Healthcare Scenario in Ambient Assisted Living*, In 5th Italian Workshop on Artificial Intelligence for Ambient Assisted Living (AIxAAL 2019) co-located with 18th International Conference of the Italian Association for Artificial Intelligence (AIxIA 2019), 2019

Others

1. Corrado Mencar, Giovanna Castellano, Ciro Castiello, Carmela Agnese De Donno, Arturo De Marinis, Corrado Fasciano, Angelo Emanuele Fiorilla, Michele Giannuzzi, Filippo Gramegna, Michele Iacobellis, Felice Iavernaro, Francesca Mazzia, Andrea Palumbo, Gino Perna, Michele Ruta, Floriano Scioscia, and Gennaro Vessio. *Drone safe-landing with*

real-time route optimization, In Booklet of abstracts of 11th EASN International Conference on Innovation in Aviation & Space to the Satisfaction of the European Citizens, 2021