PIACERE

Deliverable D5.7

IOP prototype - v1

Editor(s):	Eneko Osaba
Responsible Partner:	TECNALIA
Status-Version:	Final - v1.0
Date:	30/11/2021
Distribution level (CO, PU):	PU

Project Number:	101000162
Project Title:	PIACERE

Title of Deliverable:	IOP prototype – v1
Due Date of Delivery to the EC	30.11.2021

Workpackage responsible for the Deliverable:	WP5 - Package, release and configure Infrastructure as Code
Editor(s):	TECNALIA, 7Bulls, Polimi, SI-MPA
Contributor(s):	Eneko Osaba (TECNALIA), Diego Rosado (TECNALIA), Iñaki Etxaniz (TECNALIA), Radosław Piliszek (7BULLS)
Reviewer(s):	Bin Xiang (POLIMI)
Approved by:	All Partners
Recommended/mandatory readers:	WP3, WP5, WP6

Abstract:	The main outcome of T5.3 from M1 to M30 will be presented in this deliverable. Each deliverable will have a Technical Specification Report and a software prototype [KR9], including the explanation of the algorithms
Keyword List:	IOP, Catalogue of Infrastructural Elements, Optimization, Multi-Objective Algorithm.
Licensing information:	This work is licensed under Creative Commons Attribution-ShareAlike 3.0 Unported (CC BY-SA 3.0) http://creativecommons.org/licenses/by-sa/3.0/
Disclaimer	This document reflects only the author's views and neither Agency nor the Commission are responsible for any use that may be made of the information contained therein

Document Description

Version	Date	Modifications Introduced	
		Modification Reason	Modified by
v0.1	19/02/2021	First TOC and sections assignment.	TECNALIA
v0.2	30/09/2021	Comments and suggestions received by consortium partners.	TECNALIA
V0.8	09/11/2021	First complete version released for internal review.	TECNALIA
V0.9	26/11/2021	Addressed comments by the internal reviewer	TECNALIA
V1.0	30/11/2021	Ready for submission	TECNALIA

Table of contents

List of tables

List of figures

Terms and abbreviations

DoA	Description of Action
DOML	DevOps Modelling Language
EC	Evolutionary Computation
GA	Grant Agreement to the project
IaC	Infrastructure as Code
IOP	IaC Optimization
KPI	Key Performance Indicator
SI	Swarm Intelligence
PRC	PIACERE Runtime Controller
SW	Software
FR	Functional Requirement
NFR	Non-Functional Requirement
CSP	Cloud Service Provider

Executive Summary

This document contains the technical description of the first Infrastructure Optimization Platform (IOP) Prototype developed in the context of PIACERE project. In the specific context of PIACERE, the optimization problem to be modelled consists of having a service to be deployed and a catalogue of infrastructural elements, with the challenge of obtaining the best combination of infrastructural and resource configuration to optimally deploy the service. This outcome depends on the infrastructural elements that could compose the service and the Functional and Non-Functional Requirements involved, which are gathered by the IOP through the DOML.

Moreover, this document also introduces both state of the art sections on Optimization and Infrastructural Elements Catalogue. This document also includes sections about how to install and use these IOP and Infrastructural Elements Catalogue first prototypes, and the license under they are published. The details about their functionality and their technical aspects are described in the corresponding sections as well as the manual and the instructions to test the software.

Taking into account that there is no specific document describing the IOP architecture, the general design of it is also presented here, so that it allows the comparison to the several prototypes and its evolution. The general requirements and the functionalities are also described in this document, as well as the coverage provided by this M12 prototype, which is the original subject of the document.

Future versions of this document will present the evolution of the IOP tool, the new features that the new prototypes will provide to the developer and the technical characteristics associated to it. These versions are planned for being ready in the months 24 (in D5.8) and 30 (in D5.9) of the project.

1 Introduction

1.1 About this deliverable

This document is the first version of the *IOP protype* deliverable, which is included in the work produced by WP5 - Package, release and configure Infrastructure as code . The objective of this deliverable is to describe the main aspects of the first prototype developed of the Infrastructural Elements Catalogue and the Optimization System. In a nutshell, we will detail in this manuscript different interesting aspects such as the state of the art, implementation and installation of each of the components.

1.2 Document structure

This document is organized in 10 sections. Section 2 consists of a general description of the work made around the topics dealt in this deliverable. Sections 3 and 4 introduce several states of the art of both Catalogue of Infrastructural Elements and Automatic Optimization Tool. Following Sections 5 and 6 are devoted to presenting the implementations made for both Infrastructural Elements Catalogue and the Automatic Optimization Tool. After that, Section 7 and 8 delve into the delivery and usage of both components described in this document. Finally, Section 9 presents the conclusions and comments for the future steps until the end of the project, while Section 10 represents the references cited in this deliverable.

2 Best configuration deployments based on optimization algorithms in PIACERE

The task T5.3 coined as *Best configuration deployments based on optimization algorithms*, is devoted to the implementation of an automatic optimization tool (IOP) in charge of selecting and deploying the optimal IaC infrastructural and resource configuration based on a set of constraints (i.e. NFRs). For conducting this task, the IOP includes an infrastructural elements catalogue where their relevant characteristics are described as well as the optimizer module.

Having said this, on the one hand, the Infrastructural Elements Catalogue component is a persistence component that stores information required by different PIACERE components. As any persistence component there are two critical aspects to be covered: how the information is 1) added and 2) retrieved. Regarding the feed of information, there are three main interactions between the Catalogue and other PIACERE components, namely: 1) the GUI/IDE based on Eclipse, 2) the PIACERE Runtime Controller (PRC), and 3) the monitoring components. Regarding the usage of information, there is only one main interaction: the IOP.

On the other hand, the optimization problem formulated in PIACERE and solved by the IOP consists of having a service to be deployed, with the principal challenge of finding an optimized deployment configuration of the IaC on the appropriate infrastructural elements that best meet the predefined constraints. In this context, it is the IOP component the responsible for finding the best possible infrastructure given the input data received. This input data is provided in DOML format and will include the optimization objectives (such as the cost, performance, or availability), and the optimization requirements. Then, the IOP performs the matchmaking for the infrastructure via the execution of an optimization intelligent technique by using the information taken as input against the available infrastructure and historical data, available from the Catalogue of Infrastructural Elements.

Finally, two aspects should be considered within the IOP:

- The first one is that the problem to be optimized will be a multi-objective one, which means that it will be composed of several conflicting objectives (such as cost and performance).
- The second aspect is that two different optimizations will be conducted in the context of PIACERE: the initial deployment of the service, and the redeployment of an already running service (if the Self-Healing component decides that this is necessary).

3 Infrastructural Elements Catalogue

3.1 Infrastructural elements description and classification (SOTA)

Intermediation among Cloud Service Providers and buyers is a largely detected need when talking about multi-cloud platforms. These intermediators, among other functions, provide a single point of entry to manage multiple cloud services for business or technical purposes. The Catalogue of Infrastructural elements in PIACERE is part of this intermediation.

In the H2020 project DECIDE [1], an intermediator called ACSmI (Advanced Cloud Service meta-intermediator) is presented. It provides a cloud services store where discovery, contracting, managing and monitoring different cloud services. ACSmI allows assessing real-time verification of the cloud services non-functional properties, and also legislation compliance enforcement. Among the functionalities it provides, we can cite the registering of services. The registry of each service covers the terms for modelling the CSPs services. Based on that register, it allows the discovering and benchmarking of cloud services. ACSmI provides filters to search and indicate which are the functional (and non-functional) requirements that the services should fulfil. Thus, ACSmI searches the services in its registry and selects the most suitable services for the selected requirements. Finally, ACSmI prioritizes the set of discovered services based on the degree of fulfilment of the NFRs requested by the user. ACSmI provides some extra features that are not contemplated in PIACERE, like the legal compliance of a service (based mainly in the location of data and the service provider, and in the data security level); the automatic contracting of services with the CSP; or the calculation of costs generated by the contracted services. Another ACSmI feature that is in line to what PIACERE pretends to offer is the monitoring. This consists of monitoring the SLA (NFRs) of the service offered by the CSPs. Different metrics were measured and assessed to detect SLA violations, and in that case, raise an alert to carry out the required actions.

3.2 Gaia-X Federated Catalogue

At European level, efforts are being invested in a European federated cloud through the Gaia-X project [2]. Gaia-X aims to create a federated open data infrastructure based on European values regarding data and cloud sovereignty. The mission of Gaia-X is to design and implement a data sharing architecture that consists of common standards for data sharing, best practices, tools, and governance mechanisms. In the next paragraphs, we will describe some Gaia-X characteristics that can be relevant in the context of PIACERE.

In its architecture document [3], the concepts in the scope of Gaia-X and their relations are described, that is, the Gaia-X conceptual model, shown in Figure 1. The Gaia-X core concepts are represented in classes. The upper part of the model shows different actors of Gaia-X (highlighted in blue), while the lower part shows elements of commercial trade and the relationship to actors outside Gaia-X.

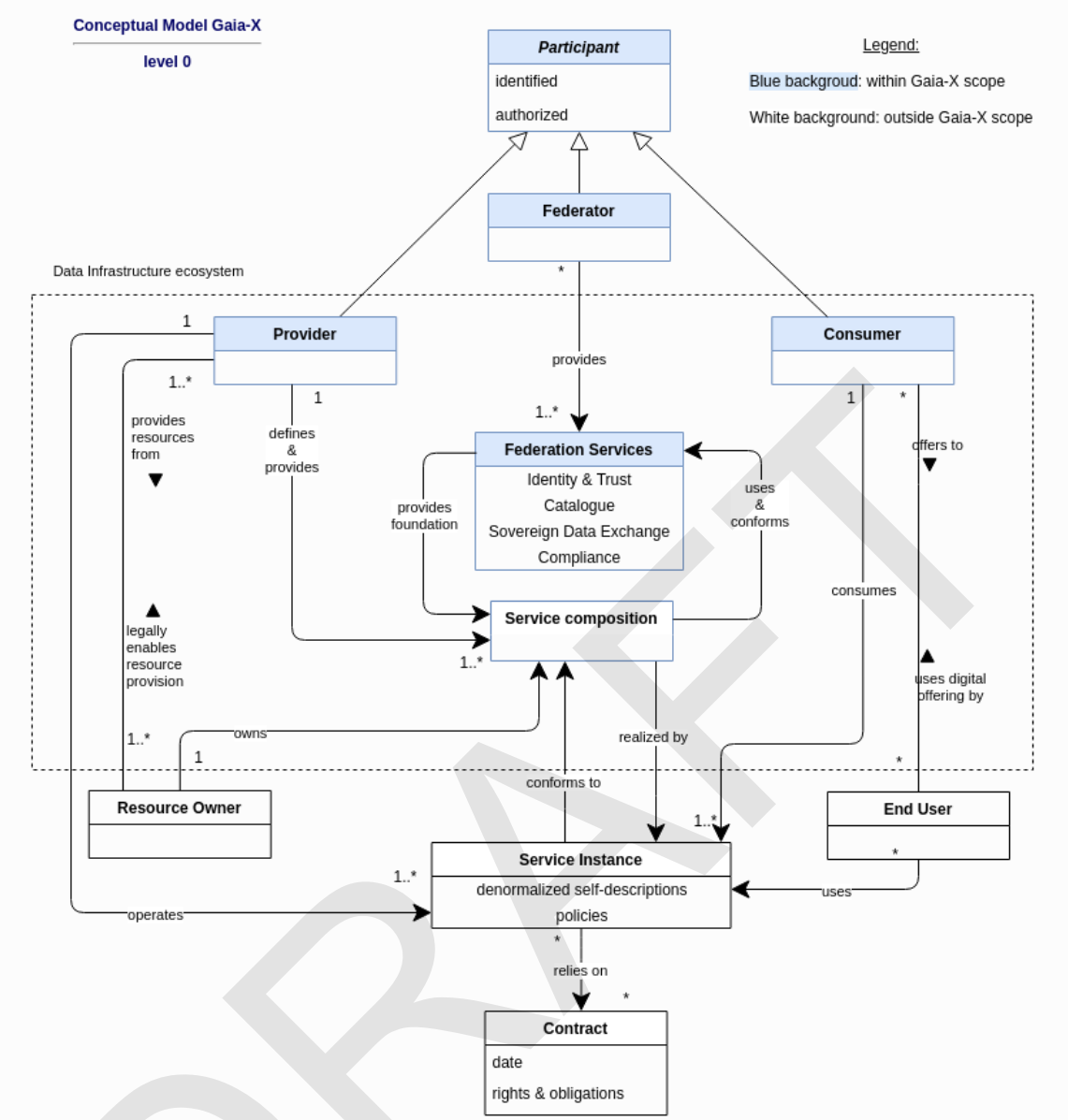

Figure 1: Gaia-X Conceptual model [3]

A *Participant* can take on one or more of the following roles: Provider, Consumer, Federator. Provider and Consumer present the core roles that are in a business-to-business relationship while the Federator enables their interaction.

- A *Provider* is a Participant who provides Resources in the Gaia-X Ecosystem. The Provider defines the Service Offering including terms and conditions as well as technical Policies. Furthermore, it provides the *Service Instance* that includes a Self-Description and associated Policies.
- A *Consumer* is a Participant who searches Service Offerings and consumes Service Instances in the Gaia-X Ecosystem to enable digital offerings for End-Users.
- *Federators* are in charge of the Federation Services and the Federation which are independent of each other. Federators are Gaia-X Participants. There can be one or more Federators per type of Federation Service.
- A *Federation* refers to a loose set of interacting actors that directly or indirectly consume, produce, or provide related Resources.

Resources and Assets describe the goods and objects of a Gaia-X Ecosystem. Resources and Assets compose the Service Offerings.

An *Asset* can be a Data Asset, a Software Asset, a Node or an Interconnection Asset. A set of Policies described in a Self-Description is bound to each Asset. The different categories of Assets are shown in Figure 2:

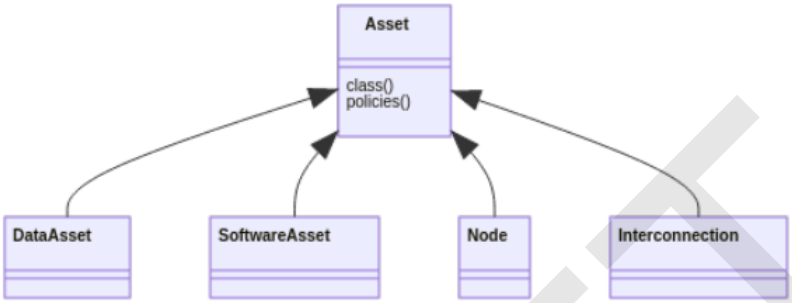

Figure 2: Assets categories [3]

- A *Data Asset* is an Asset that consists of data in any form and necessary information for data sharing.
- A *Node* is an Asset and represents a computational or physical entity that hosts, manipulates, or interacts with other computational or physical entities.
- A *Software Asset* is a form of Assets that consist of non-physical functions.
- An *Interconnection* as an Asset presents the connection between two or more Nodes. These Nodes are usually deployed in different infrastructure domains and owned by different stakeholders, such as Consumers and/or Providers. The Interconnection between the Nodes can be seen as a path which exhibits special characteristics, such as latency, bandwidth and security guarantees, that go beyond the characteristics of a path over the public Internet.

The difference between *Resources* and *Assets* lies in that Resources represent those elements necessary to supply Assets. In other words, they are internal Service Instances not available for order. For example, the running instance that provides a data set is a Resource.

Federation Services are services required for the operational implementation of a Gaia-X Data Ecosystem. They comprise four groups of services that are necessary to enable Federation of Resources, Participants and interactions between Ecosystems. The four service groups are Identity and Trust, Federated Catalogue, Sovereign Data Exchange and Compliance. The Federation Services provide the foundation for Service Offerings.

A *Service Offering* is defined as a set of Resources that a Provider aggregates and publishes as a single entry in a Catalogue. Service Offerings may themselves be aggregated realizing service composition. The instantiation of a Service Offering is the deliverable of a Provider to a Consumer.

Federated Catalogue

The *Federated Catalogue* [3] constitutes an indexed repository of Gaia-X Self-Descriptions to enable the discovery and selection of Providers and their Service Offerings. The Self-Descriptions are the properties and Claims of Participants and Resources, representing key elements of transparency and trust in Gaia-X.

Self-Descriptions intended for public usage can be published in a Catalogue where they can be found by potential Consumers. The goal of Catalogues in Gaia-X is to enable Consumers to find best-matching offerings and to monitor for relevant changes of the offerings. The Providers decide in a self-sovereign manner which information they want to make public in a Catalogue and which information they only want to share privately.

A Catalogue stores Self-Descriptions both as standalone and as aggregated in a graph data structure. The Self-Description Storage contains the raw published Self-Description files (in the JSON-LD format). The individual Self-Descriptions can reference each other. The Self-Description Graph is the basis for advanced query mechanisms that consider the references between and among Self-Descriptions [3].

Ecosystem-specific Catalogues (e.g., for the automotive domain) and even company-internal Catalogues (with private Self-Descriptions to be used only internally) can be linked to the system of federated Catalogues. The Catalogue federation is used to exchange relevant Self-Descriptions and updates thereof. It is not used to execute queries in a distributed fashion [3].

The system of Federated Catalogues consists of a top-level Catalogue operated by Gaia-X and provides the means to link to Ecosystem-specific Catalogues (e.g., for the automotive domain) and even company-internal Catalogues with private Self-Descriptions to be used only internally. Self-Descriptions in a Catalogue are either loaded directly into a Catalogue or exchanged from another Catalogue through inter-Catalogue synchronization functions [3].

Since Self-Descriptions are protected by cryptographic signatures, they are immutable and cannot be changed once published. This implies that after any changes to a Self-Description, the Participant as the Self-Description issuer has to sign the Self-Description again and release it as a new version.

The possible states for the Self-Description lifecycle are four. All states except “Active” are terminal, which means that no further state transitions are allowed. The states are [3] [4]:

- **Active**: is the default state.
- **End-of-Life**: after a timeout date, e.g., the expiry of a cryptographic signature.
- **Deprecated**: by a newer Self-Description.
- **Revoked**: by the original issuer or a trusted party (e.g. because it contained wrong or fraudulent information).

The Self-Description Graph contains the information imported from the Self-Descriptions that are known to a Catalogue and in an "Active" state [4]. The Self-Description Graph allows for complex queries across Self-Descriptions. To present search results objectively and without discrimination, compliant Catalogues use a query engine with no internal ranking of results. Users can define filters and sort-criteria in their queries. But if some results have no unique ordering according to the defined sort-criteria, they are randomized. Self-Descriptions can be communicated to the Catalogue by third parties, as the trust verification is independent of the distribution mechanism.

Self-Descriptions can be marked by the issuer as "private" to prevent them from being copied to a public Catalogue by a third party that received the Self-Description over a private channel [3]. The Catalogues have no built-in user interface, instead provide an API that can be used by an external user interface or technical clients [4]. The interfaces of the Gaia-X Federation Services use REST and the OpenAPI specification [5] to describe them.

A *Visitor* is an anonymous user accessing a Catalogue without a known account. Every Non-Visitor user interacts with a Catalogue REST API in the context of a session. Another option to interact with a Catalogue is to use a GUI frontend (e.g., a Gaia-X Portal or a custom GUI implementation) that uses a Catalogue REST API in the background. The interaction between a Catalogue and its GUI frontend is based on an authenticated session for the individual user of the GUI frontend.

Gaia-X Self-Descriptions express characteristics of Resources, Service Offerings and Participants that are linked to their respective Identifiers. Providers are responsible for the creation of Self-Descriptions of their Resources. In addition to self-declared Claims made by Participants, a Self-Description may comprise Credentials issued and signed by trusted parties. Self-Descriptions can be used for [3]:

- Discovery and composition of Service Offerings in a Catalogue
- Tool-assisted evaluation, integration and orchestration of Service Instances/Resources
- Enforcement, continuous validation and trust monitoring
- Negotiation of contractual terms concerning Resources of a Service Offering and Participants

Gaia-X Self-Descriptions are characterized by the following properties [3]:

- Machine-readable and machine-interpretable
- Technology-agnostic
- Adhering to a generalized schema with expressive semantics and validation rules
- Interoperable, following standards in terms of format, structure, and included expressions (semantics)
- Flexible, extensible and future-proof in that new properties can be easily added
- Navigable and referenceable from anywhere in a decentralized fashion
- Accompanied by statements of proof (e.g., certificates or signatures), making them cryptographically trustworthy

The exchange format for Self-Descriptions is JSON-LD. JSON-LD uses JSON encoding to represent subject-predicate-object triples according to the W3C Resource Description Framework (RDF). The relations between Self-Descriptions form a graph with typed edges, which is called the Self-Description Graph. The Catalogues implement a query algorithm on top of the Self-Description Graph [3].

To foster interoperability, Self-Description schemas with optional and mandatory properties and relations are defined. A Self-Description has to state which schemas are used in its metadata. A Self-Description schema corresponds to a class in RDF. The Self-Description schemas form an extensible class hierarchy with inheritance [3].

3.3 Cloud Service Brokers

Cloud Service Brokerage (CSB) is being referred to as intermediate individuals connecting Cloud Service Providers and buyers that help customize and improve business data on multi-cloud platforms. Cloud brokers provide a single point of entry to manage multiple cloud services for business or technical purposes. Their goal is to integrate or aggregate services, to enhance their security, or to do anything which adds a significant layer of value to the original cloud services being offered.

We could define a cross-cloud application as one that consumes more than one cloud API, from different CSPs, under a single version of the application. Most cloud brokers offer a cross-cloud architecture solution, therefore in this analysis we are using the term CSB implying also a cross-

cloud architecture service that offers an administrative console that can be used to manage resources and applications across different CSPs.

As is indicated in [6], cloud brokerage services are constantly enhanced to fulfil business requirements like those listed below, but for now the market seems missing a solution that integrates all of them:

- Support monitoring solutions. Brokers that provide the solutions include RackSpace [7], Jamcracker [8], Computenext [9], CloudBolt [10], Embotics [11], Morpheus [12] and others. As for monitoring tools that can be used for monitoring purposes, there are the following options:
  - Software as a Service (SaaS): AppDynamics [13], BMC's TrueSight Pulse Monitoring-as-a-Service [14], New Relic [15].
  - Stand-alone packages: Nagios [16], Zabbix [17].
  - Services provided by the platform hosting the application: OpenStack Monasca [18], AWS CloudWatch [19].
- Support application adaptation. For example, Jamcracker supports VM sprawl management, using policy-based tools to automate high-scale provisioning and de-provisioning of resources [20].
- Allow the best deployment and operation options using a smart decision system. Examples of cloud brokers that support smart decision engines would be T-Systems [21] and Nephos Technologies [22].
- Offer broad range of cloud services, or cloud marketplaces. Brokers that provide these solutions include Computenext [9], UberCloud [23], Cloudmore [24], CloudBolt [10], Embotics [11], Morpheus [12].

A comparative analysis has been carried out in [25] to compare Cloud Service Brokers in the market, based on the functionality offered by ACSmI, the Cloud Services Intermediator defined in the DECIDE [1] project. In the analysis, several solutions are studied, which are listed below:

Cloudmore

Cloudmore [26] provides a platform to procure, deploy and consume IT services. It is designed to support different roles such as administrators, managers and end-users.

The Cloudmore cloud broker enables "*the internal or external IT function to set up a service catalogue, business rules, security and a delegated administration structure in addition to having an aggregated view of financial data, use and utilization*". Among its features, it supports aggregation of services and its corresponding bills and contract management, maintains an event log and a customizable e-store of services.

Cloudmore integrates the following services: Microsoft Direct - Office 365 and related services, Microsoft Direct – Azure, IBM -IBM Spectrum Protect (previously Tivoli Storage Manager), VMWare – vCloud, Microsoft - Hosted Exchange, LiveDrive - Live Vault Server Backup, HP Connected - Desktop backup, and F-Secure - End point protection.

IBM Cloud Brokerage solutions

IBM Cloud Brokerage solutions [27] support companies in the planning, acquisition and management IT resources across different cloud models from multiple suppliers while reducing compliance risk and overall IT costs. This solution is based on Gravitant, a cloud service broker acquired by IBM in 2015.

It offers four services: (i) IBM Cloud Brokerage Managed Services - Cost and Asset Management, to management the costs of the cloud resources; (ii) IBM Integrated Managed Infrastructure Services: to manage the hybrid infrastructure; (iii) IBM Cloud Brokerage Workload Planning: to understand the costs before the actual deployment; (iv) IBM Cloud Brokerage Managed Services - Store: to search and contract cloud services.

Nephos technologies cloud management

Nephos technologies cloud management [28] provides a centralized cloud management solution, where users can have visibility of the usage and costs of their cloud infrastructure. It is integrated with Continuous Delivery (CD) tools such as Puppet or Chef.

Intercloud

Intercloud platform [29] is an "*application-aware platform offering private access to any cloud provider, securely and efficiently delivering cloud applications wherever they may be required*". It works under a private network, and it integrates CSPs such as Amazon, Oracle, Alibaba, Azure, Google Cloud and IBM Cloud.

Jamcracker

Jamcracker [30] is a platform that provides cloud services brokerage, cloud governance and cloud services management which enables organizations to create, deliver, and manage multi-cloud services.

Among its features, Jamcracker has a service catalogue and supports provisioning, allowing the user to manage the costs and the budget, and monitoring the workloads and resource utilization. It integrates CSPs such as AWS, Azure, Google Cloud, VMWare Cloud, IBM Cloud, Office 356 and Gsuite.

CloudBolt

CloudBolt [10] is a hybrid cloud platform that aims to be a central management point seeking the aggregation of operational data, service lifecycle management and orchestration and reporting. CloudBolt supports a variety of cloud technologies, from virtualization to public and hybrid clouds. More specifically, it supports AWS, Microsoft Azure Stack, Citrix XenServer, Microsoft Hyper-V, Nutanix Acropolis, KVM/QEMU, OpenStack, Red Hat Enterprise Virtualization, VMware vCenter, CenturyLink Cloud, Google Cloud Platform, IBM SoftLayer, Microsoft Azure/AzureRM and Oracle Cloud. For storage, CloudBolt only supports Tintri. To manage the configuration of the infrastructure, CloudBolt supports Chef, Ansible and Puppet. Containers are supported but only Kubernetes.

CloudBolt is offered under a Free trial business model in which the period for testing the product is 14 days. On top of that, CloudBolt follows a partner-based program.

Embotics

Embotics Snow Commander Hybrid Cloud Management platform [11] is a solution that aims to support multiple hypervisors and clouds. It allows to deploy and automate provisioning across virtualized, private and public cloud infrastructures while optimizing costs and resources.

Snow Commander orchestrates and manages the following environments VMware vSphere, Microsoft Hyper-V, Amazon Web Services (AWS) and Microsoft Azure. This product supports currently Docker containers. The version 2.0 supports Docker Swarm and Kubernetes. Embotics

states that they can also integrate some CI/CD tools such as Git, Jenkins, Puppet and Chef in their Snow Commander Workflow.

The analysis shown next is based in comparing the solutions among them in relation to a set of functionalities that an Intermediator shall offer. This set of functionalities is listed following:

- F1: offers mechanisms to authorize and manage different roles and profiles that can access, namely developers / operators, CSPs, legal expert. The UI shall differ depending on who is accessing it.
- F2: offers the possibility to CSPs to endorse their services by inserting the required information such as complied-with certification schemes and certain metrics for availability and performance.
- F3: An operator may manually verify that all the information that the CSPs have included is valid.
- F4: offers a dashboard to CSPs where they can visualize the status of their services and information such as the metrics monitored.
- F5: provides different criterion to discover a set of services that match those characteristics.
- F6: Shall assess (and rank) the services that match the required criterion. This classification of services will be returned to the user.
- F7: contracts the selected service offerings, either automatically or manually.
- F8: offers mechanisms to deploy the multi-cloud application on the selected resources.
- F9: allows monitoring the NFR metrics, raising a violation whenever a CSLO is not being fulfilled. Such non-compliance shall be stored as it may affect future re-deployments.
- F10: allows to bill the cloud service offerings to the developer / operator.

The following table show the comparative analysis of the above tools with respect to the previously listed set of functionalities.

Table 1: Comparative analysis of the above tools with respect to the previously listed set of functionalities (source: [25])

ID	Cloudmore	Cloud Brokerage solutions	Nephos tech. cloud management	Intercloud	Jamcracker	CloudBolt	Embotic
F1	Fully	Fully	Fully	Fully	Fully	Fully	Fully
F2	Fully	Fully	Fully	Fully	Fully, through partnerships	Fully, through partnerships	Partially
F3	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown
F4	Partially	Partially	Partially	Partially	Partially	Not covered	Not covered
F5	Partially (not focused on NFRs)	Partially (not focused on NFRs but on costs)	Not covered	Not covered	Partially (not focused on NFRs but on costs)	Partially (not focused on NFRs but on costs)	Not covered. The developer shall select the services where the application will be deployed
F6	Partially (not focused on NFRs)	Partially (not focused on NFRs)	Not covered	Not covered	Partially (not focused on NFRs)	Partially (not focused on NFRs but on costs)	Not covered. The developer shall select the services where the application will be deployed
F7	Partially	Partially	Partially	Partially	Partially	Partially	Partially
F8	Not covered	Not covered	Partially (through Puppet or Chef)	Not covered	Not covered	Partially (through Puppet or Chef)	Partially (through Puppet or Chef)
F9	Not covered	Not covered	Not covered	Not covered	Not covered	Partially. Only costs are monitored	Partially. Only costs are monitored
F10	Fully	Fully	Fully	Fully	Fully	Fully	Fully

4 Automatic Optimization Tool (IoP)

4.1 Optimization (SOTA)

Optimization is one of the most studied areas in artificial intelligence. A plethora of works are produced yearly focused on the solving of many diverse problems of this kind (infrastructure optimization) by using to a vast spectrum of techniques. Many different optimization problems can be found in the literature depending on their main characteristics and their optimization variables. Thus, we can detect optimization types as linear or nonlinear, continuous or combinatorial, among other. Efficiently dealing such problems usually demands huge computational resources, especially when the problem to be solved represents a complex real-world situation with a significant number of constraints or variables. In this way, and because the inherent utility of optimization solvers, a wide spectrum of solving approaches have been formulated by the related community over the last years for their application to these problems.

The mathematical formulation and tackling of optimization problems by means of intelligent algorithms has gained a significant popularity. Furthermore, this situation has been strengthened in the last years due to the appearance of new problem modelling paradigms (large scale optimization [31], multi-objective problems [32], etc.), and the burst of subfields as Evolutionary Computation (EC) and Swarm Intelligence (SI) [33] [34]. In this regard, and since the social and scientific interests and value related to the resolution of optimization problems, a myriad of techniques has been modelled and developed in last years for being adapted to the huge variety of use cases. Most frequently used approaches can be categorized in three different groups: exact methods [35], heuristics [36] and metaheuristics [37]. In the context of this project, metaheuristics will be used. In few words, a metaheuristic is an optimization method which tackles a specific problem employing only general knowledge common to a wide variety of problems with similar features. In this way, metaheuristics widely explore the space of solutions with the intention of reaching good results with independence of the problem. For this reason, metaheuristics are considered as appropriate to deal with real-world problems with highly complex mathematical formulations, since the efficiency explore the solution space regardless of the problem [38]. For this reason, and thanks to their adaptability and efficiency, metaheuristics are enjoying a greater popularity [39] in comparison to exact methods and heuristics.

Among the vast spectrum of optimization types, we can find valuable paradigms and formulations, such as Dynamic Optimization, Transfer Optimization, Robust Optimization, or many others [40] [41] [42]. In PIACERE project, we specifically focus our attention on multi-objective optimization, which will be briefly described in next subsection.

4.1.1 Multi-objective Optimization

In a nutshell, the principal goal of multi-objective optimization (MOO) is to find a group of solutions that balance between several conflicting objectives, which are defined on a single domain (and consequently, on a sole search space). Thus, MOO assumes a Pareto trade-off among these conflicting objectives, for which the implemented MOO algorithms produce an estimation in the form of a group of solutions. For this reason, there is not a *unique* solution to each MOO problem. On the contrary, different solutions that optimize each objective to a certain degree exist.

In this context, solvers framed in the wide category of EC have gained an ever-growing popularity also for solving MOO problems, with a significant amount of published works focused on the adaptation of different EC approaches to MOO problems. The principal reason that explains the momentum acquired by this family of techniques lies in their good performance, which has been scientifically demonstrated in hundreds of research topics and real-world scenarios. In this

regard, different inspirational sources have been embraced for developing optimization algorithms (giving rise to many bio-inspired metaheuristics), such as the behavioural patterns of ants, birds, bat or even buffalos.

In recent years, most of these evolutionary techniques have been efficiently applied to a wide variety of topics. To cite a few, the well-known Bat Algorithm has been adapted for give an answer to situations related to energy [43], sports training planning [44] or transportation [45] [46], whereas the Firefly Algorithm has been used to problems appeared in medicine [47], industry [48] or logistics [49] [50]. This is an extremely reduced yet exemplary sample of the heterogeneous activities behind this field. A deeper analysis of the related literature in this topic could be highly extensive. This is the reason why many comprehensive survey papers have been contributed to demonstrate the huge literature behind certain algorithms. Nonetheless, Ant Colony Optimization and Genetic Algorithms are the most widely used algorithms of this type, with recent papers focused on these both approaches [51] [52].

With all this, the choosing of an algorithm for dealing with an optimization problem is a difficult task to conduct. Usually, this election depends on many factors, such as the amount of optimization objectives, the complexity of the fitness function or the non-functional requirements (understandability of the method, time of response…). Anyway, some well-stablished MOO techniques exist in the literature, which should arguably influence the final decision, inspiring the final implementation of the deploying algorithm. In this subsection, we outline some of these algorithms, which should be carefully analysed in the context of PIACERE IOP platform.

4.1.1.1 Genetic Algorithm based MOO methods

The following are methods mainly based on the well-known Genetic Algorithm, which is arguably the most used EC scheme for solving MOO problems.

- *Strength Pareto evolutionary algorithm 2 (SPEA2)*: SPEA2 has some important positive points that can be listed as follows: i) a density estimation method based on the nearest neighbor is considered, which permits a precise guidance of the search procedure; ii) an enhanced fitness assignment scheme, in which for each individual of the population, the algorithms consider how many individuals it dominates and it is dominated by; and iii) an archive truncation mechanism that assures the retention of boundary solutions [53].

- *Non-dominated sorting genetic algorithm II (NSGA-II):* this is, arguably, the most well-known and used MOO method, which consists of a generational genetic algorithm which resorts to a Pareto ranking scheme to enhance the convergence to the Pareto front. NSGA-II algorithm also employs the crowding distance density estimator for preserving the diversity on the front [54].

- *Multi-objective evolutionary algorithm based on decomposition (MOEA/D):* the main basis of this algorithm is the decomposition of the MOO problem into a different scalar optimizations subproblems, which then tackles them in a simultaneous way. Each of the generated subproblems is optimized by employing information drawn from its neighboring subproblems. This last feature makes MOEA/D algorithm an efficient approach in terms of computational complexity [55].

- *S metric selection evolutionary multi-objective optimization algorithm (SMS-EMOA):* this technique employs the concept of non-dominated sorting in combination with a selection operator based on the hypervolume measure. The population of individuals

evolves to a distributed set of solutions, maintaining the diversity of the population, and also focusing on potentially good regions of the Pareto front [56].

- *Multiple Trajectory Search (MTS):* the first step of the MTS is the generation of a uniformly distributed group of solutions. After that, this set is separated into foreground and background solutions. The optimization search process is mainly focused on foreground solutions, and partly on the background ones. The algorithm selects and applies local search procedures on solutions in an iterative way. Thanks to a coined as size-varied neighborhood searches, the algorithm effectively solves MOO problems [57].

- *MOCell*: this MOO technique can be categorized within the cellular genetic algorithm family. MOCell employs an external archive for maintaining non-dominated solutions, and a feedback function in which solutions from this archive are employed in the selection step of the method [58].

4.1.1.2 Differential Algorithm-based approaches

Being also one of the most famous and recognized EC method, Differential Evolution has also served as inspiration for several promising MOO solvers.

- *MOEA/D based on differential evolution (MOEA/D-DE)*: MOEA/D-DE is basically a variant of MOEA/D in which the Genetic Algorithm operator is replaced by the differential evolution (DE) operator [59].

- *Generalized differential evolution 3 (GDE)*: this is a technique extended from the DE and get by modifying the selection rule of the basic DE. The main concept of the GDE with the selection rule is that the trial vector is used for replacing the old one in the subsequent generation, only if it weakly constraint-dominates the old vector [60].

4.1.1.3 Particle Swarm Optimization-based Algorithms

Particle Swarm Optimization (PSO) is arguably the most reputed method of the knowledge branch known as Swarm Intelligence. The demonstrated efficiency of this technique has led researchers and practitioners to propose different MOO solvers using as inspiration this successful algorithm.

- *Multi-objective particle swarm optimization (MOPSO):* this method employs the mechanism of Pareto dominance for establishing the flight direction of a particle. Furthermore, this method maintains previously encountered non-dominated vectors in a repository that it is employed by other particles of the population to guide their flight [61].

- *Speed-constrained multi-objective particle swarm optimization (SMPSO):* the main feature of this method is the adoption of a velocity constraint function, with the objective of avoiding particles flying outside the search space. The SMPSO resorts to an external archive for saving the non-dominated solutions found, from which the best particle is also chosen [62].

- *Decomposition-based particle swarm optimization (dMOPSO):* this algorithm updates the position of each individual employing a group of solutions deemed as the global best following a decomposition approach. This algorithm is mainly featured by using a memory reinitialization mechanism which provides diversity to the population [63].

4.1.1.4 Memetic-based Algorithms

Since the initial steps of the EC research field, practitioners have dedicated some efforts on combining the advantages of techniques and mechanisms into single solvers with the hope of overcoming the inherent disadvantages shown by off-the-shelf techniques. In this regard, Memetic Algorithms (MA) resort to this design principle by analyzing potential synergies among evolutionary frameworks with local search mechanisms. As a result of this research trend, practitioners have also developed interesting MOO methods that can be framed in this category, such as the well-known *Memetic algorithm based on Pareto archived evolution strategy (M-PAES).* This technique uses the efficient local search strategy employed in the Pareto archived evolution strategy (PAES) and hybridizes it with the consideration of a population and recombination [64].

4.1.1.5 Many-objective algorithms

Historically, multi-objective problems are conceived for optimizing two different conflicting objectives. Anyway, in many real-world situations, the number of objectives to be optimized can be higher. This trend gives rise to a sub-branch known as many-objective algorithms, which must deal with the existence of many conflicting objectives, trying also to define the relationships and synergies among them. These are some interesting alternatives to consider:

- *Non-dominated Sorting Genetic Algorithm III (NSGA-III)*: the NSGA-III is the evolved version of the reputed NSGA-II, which follows a similar structure. NSGA-III puts an special emphasizes on non-dominated population of solutions, yet close to a group of supplied reference points [65].

- *Strength Pareto evolutionary algorithm based on reference direction (SPEA/R)*: the SPEA/R is also an enhanced version of a classic MOO algorithm: the SPEA. In this case, SPEA/R introduces some novel mechanisms: i) a reference direction-based density estimator, ii) a novel environmental selection strategy, and iii) a new fitness assignment procedure [66].

- *Multi-objective evolutionary algorithm based on dominance and decomposition (MOEA/DD)*: this is a unified method, which works on the combination of decomposition- and dominance-based approaches [67].

- *Hypervolume-based estimation algorithm (HypE)*: the HypE is a fast search method which resorts to the Monte Carlo simulation mechanism to approximate the exact hypervolume values. The principal objective is not that the indicator values are crucial, but the rankings of solutions induced by the hypervolume [68].

- *Many-Objective Metaheuristic Based on the R2 Indicator (MOMBI2)*: the main inspiration of this method is the using of the R2 indicator. This R2 indicator is strictly correlated with the well-known hypervolume, demonstrating some advantages in comparison, such as an efficient computational cost [69].

4.1.1.6 Other methods to be considered

The algorithms already highlighted in this document are conceived for solving multi-objective and/or many objective problems. In any case, it could be possible to embrace other solving approaches for the resolution of the optimization problem, in case above mentioned approaches are not able to meet functional or non-functional requirements:

- *Micro algorithms*: these metaheuristics are conceived for solving optimization problems in a very fast way, using the minimum required computational resources. The most representative example of this class is the Micro-Genetic Algorithm [70].

- *Transfer Optimization*: Transfer Optimization is a new research area within optimization [41]. The main idea behind this concept is to exploit what has been learned when optimizing one optimization task, toward facing another related or unrelated problem.

- *Single objective algorithms*: These methods are the most employed ones for dealing with optimization tasks. In the context of PIACERE, these algorithms can be considered in case the time of response is critical. In this case, we can use a single objective algorithm with a weight-based objective function. In this category, we have several alternatives such as local-search methods such as Tabu Search or Simulated Annealing. We can algo contemplate evolutionary methods such as Genetic Algorithm of Differential Evolution. Finally, we can use more sophisticated Swarm Intelligence techniques such as Bat Algorithm or Firefly Algorithm.

4.1.2 Multi-objective solving frameworks

We show in Table 2 a summary of the main features of a representative set of metaheuristic optimization frameworks. Considering the proven strength of these frameworks, as part of PIACERE project an experimentation will be made comparing the performance of algorithms drawn from some of these frameworks. That is, the objective of the task is not just to decide which is the best algorithm for the use cases, but also the most efficient framework to implement it. We depict in this table the programming language, the main objective of the framework (SOO: single objective optimization, MOO: multi-objective optimization), the current version, and last update date.

Table 2 Main features of representative multi-objective optimization frameworks.

Framework	Language	Algorithms	Current Version	Last Update
ECJ	Java	**SOO**/MOO	27	August 2019
HeuristicLab	C#	**SOO**/MOO	3.3	July 2019
jMetal	Java	SOO/**MOO**	5.1.1	June 2021
jMetalPy	Python	SOO/**MOO**	1.5.3	February 2020
MOEAFramework	Java	SOO/**MOO**	2.13	December 2019
NiaPy	Python	SOO	2.0.0	November 2019
Pagmo	C++	**SOO**/MOO	2.18.0	August 2021
PlatEMO	MATLAB	MOO	3.3	August 2021
Pygmo	Python	**SOO**/MOO	2.18	June 2021

If we focus our attention on the programming language feature, we can observe how Java and Python are the most use ones, finding also cases such as HeuristicLab, Pagmo and PlatEMO, which are developed in C#, C++ and MATLAB, respectively. We can assume, for example, that Python-based frameworks would be computationally inefficient. For this reason, if this feature is a non-functional requirement, theirs based on C++ or Java could be more appropriate. Anyway, Pygmo is in fact based on Pagmo (it is basically a Python wrapper of that framework), so it can be considered as competitive alternative in terms of performance. The rest of frameworks implemented in Python are considerably slower. For example, if we deem jMetalPy (Python) and jMetal (Java), it can be observed that executing the same algorithm with identical configuration can take up to fourteen times more running time in Python than in Java. Anyway, the advantages that Python provides for fast prototyping and the significant number of libraries

for data analysis and make frameworks implemented in this language to be ideal for testing and fine-tuning.

Also, in Table 1, we have spotlight if a framework offers both types of methods, but it is more dedicated to one of them. The specialization of the library on single-objective or MOO can also be a good reason for selecting a framework. In this way, if the problem to be solved is single-objective, alternatives such as ECJ, Pygmo, or NiaPy provide a wide range of functionalities and methods to face it. This same philosophy applies to the other alternatives concerning MOO.

4.1.3 SOTA related to PIACERE project

In the specific context of PIACERE, the optimization problem to be modelled consists of having a service to be deployed and a catalogue of infrastructural elements, with the challenge of obtaining the best combination of infrastructural and resource configuration to optimally deploy the service. Despite being an incipient research field, some studies can be found in the literature dealing with similar problems.

In [71], authors built an optimization technique based on the well-known NSGA-II, which matches the requirements of a certain Big Data application to the capabilities offered by an IaaS infrastructure and the Big Data platform deployed therein. This study also explores the Pareto-optimal frontier among three design concepts when defining an IaaS infrastructure for Big Data applications: net computing capacity, cost, and reliability.

From this pioneering work and inspired by the same philosophy, [72] optimizes two principal objectives: the meeting of the microservices' non-functional, and the fulfilment of the features group by the developers of these microservices.

These works are specially inspiring for this project, the synergy between Big Data and Cloud Computing encounters in IaC a practical Cloud model, by which practitioners can deploy Big Data functionalities externally in an efficient fashion and with independence of the service provider [73]. In this way, instead of buying hardware, users can buy IaaS usage time on demand depending on their specific needs. This is a similar concept than the electricity or other utility billing [74].

Finally, it is noteworthy that the literature is scarce in papers focused on the optimal selection of infrastructural elements for providing a specific service. A slightly related example can be found in [75]. This work proposed an automated platform for answering the sizing cluster question considering several functionalities such as tailored cluster resources, task scheduling properties, configuration settings or the performance prediction. The main difference is that the system presented in that work does not offer a unique solution to infrastructure definition. On the contrary, it assists the user giving a response to the most common doubts.

Furthermore, in a related H2020 project known as MELODIC, a novel, multi-cloud application deployment optimization engine based on user-provided utility function was proposed [76]. The engine works continuously at run-time, as opposed to being one-off before the application gets deployed, and thus ensures the application is always running in the optimal way. The optimization problem (via utility functions) is modeled in CAMEL (Cloud Application Modelling Language), similar to TOSCA (Topology and Orchestration Specification for Cloud Applications). The model is translated to a Constraint Programming model (CP model) which allows for applying known optimization algorithms. The optimization is triggered by re-evaluating the utility functions as time progresses and new data is available from the monitoring system. Several solvers work together to find the best solution.

To finish with this section, it is proper to mention that a considerable research works regarding the cloud resource management can be found in the literature. Dewangan et al. [77] present an extensive analysis of different resource provisioning systems. They highlight the classifications of resource management techniques, objective functions, and open research challenges and issues while analyzing resource management techniques. Khattar et al. [78] provide a survey in the domain of energy efficiency in cloud computing, which classifies heuristics-based optimization methods and the dynamic power management techniques. Lastly, we highlight in this section four different related concepts and some interesting backgrounds.

- **Cloud resource scheduling:** Zhu et al. [79] propose a self-adapting task scheduling algorithm using learning automata model for container cloud. They design a reward-penalty mechanism for scheduling actions considering the states of resources and tasks and optimize the selection of actions. A framework of task load monitoring with buffer queue is proposed to achieve dynamic scheduling based on priority. In [80], Zhu et al. build the resource model of the data center and the dynamic power model of the physical machine, and then propose a three-dimensional virtual resource scheduling method for energy saving. Zhao et al. [81] provide optimization solutions for Analytics-as-a-Service (AaaS) platforms that automatically and elastically provision cloud resources to execute queries guaranteeing Service Level Agreements across a range of Quality-of-Service requirements. Admission control and resource scheduling algorithms are proposed for AaaS platforms to maximize profits while providing time-minimized query execution plans to meet user demands and expectations.
- **Cloud resource configuration:** In [82], Ciavotta et al. tackle the problem of supporting the design-time analysis of Cloud applications to identify a cost-optimized component allocation onto Virtual Machine (VM) services, taking performance requirements into account. They present an approach and a tool to support users in modeling the architecture of an application, in defining performance requirements and deployment constraints, and then in mapping each component into a corresponding VM. In [83], Mireslami et al. propose a hybrid method to allocate cloud resources according to the dynamic user demands by developing a two-phase algorithm including reservation and dynamic provision. They minimize the total deployment cost by formulating each phase as an optimization problem satisfying quality of service. Due to the uncertainty of cloud demands, they develop a stochastic optimization approach by modeling user demands as random variables. Gianniti et al. [84] study the rightsizing of Cloud deployed clusters. They propose a tool implementing a parallel and distributed simulation-optimization technique to explore the cloud configurations to minimize deployment cost under quality-of-service constraints. In [85], Mann et al. argue that the optimization problems of resource provisioning in virtualized environments on mapping virtual machines (VMs) to physical machines or mapping application components to VMs influence each other significantly. They propose a formulation for the joint optimization of the two mappings, taking into account sizing aspects, colocation constraints, license costs, and hardware affinity relations. Osypanka et al. [86] present an approach using anomaly detection, machine learning and particle swarm optimization to achieve a cost-optimal cloud resource configuration. It works in a closed loop without the need for external supervision or initialization and builds knowledge about the usage patterns of the system being optimized and filters out anomalous situations on the fly. Mireslami et al. [87] present a cost-effective and runtime friendly algorithm that minimizes the cost of deploying a Web application in cloud environment while meeting the QoS performance requirements.
- **Cloud resource allocation:** Xiang et al. [88] propose a mathematical model to perform a joint slicing of mobile network and edge computation resources, aiming at minimizing the total latency of transmitting, outsourcing and processing user traffic, under the

constraint of user tolerable latency for multiple classes of traffic. They formulate it as a mixed-integer nonlinear programming (MINLP) problem. Li et al. [89] propose the resource optimization and load balancing model for edge cloud considering factors such as user preferences, SLA and cost. A data migration strategy is designed by considering load balance while migrating jobs to new resources. In [90], Li et al. formulate resource reservation and allocation in mobile cloud computing with uncertain demands of mobile users as a robust optimization model. They propose a robust joint resource reservation and allocation algorithm to provision the radio resources and the virtual machine resources. Alam et al. [91] propose a resource allocation approach for cloud computing with the aim to maximize reliability while minimizing the cost.

- **Cloud resource placement (VNF specific):** In [92], Yue et al. focus on the virtual network function (VNF) placement problem for mapping service function chain (SFC) requests in networks, considering the delay requirement of requests. They formulate the problem as an integer linear programming (ILP) model to minimize the total resource consumption. Xu et al. [93] consider provisioning network services in an NFV-enabled network that consists of data centers for implementing VNF instances of service chains and switches. They study the throughput maximization problem with the aim to admit as many user requests as possible while minimizing the implementation cost of the requests, assuming that limited numbers of instances of each service chain have been stored in data centers. In [94], Li et al. address the virtual network function (VNF) placement problem in cloud datacenter considering users' service function chain requests with the time-varying workloads. The problem is formulated as an integer linear programming (ILP) model with the aim of minimizing the number of used physical machines.

4.2 Optimization in PIACERE

In this section, we contextualize the optimization problem that we tackle in the frame of PIACERE. To do that, first, we describe the modelled problem in Section 5.2.1, showing some examples that help to easy understand how the developed optimization methods will be able to find promising solutions in the search space. After that, in Section 5.2.2, we briefly describe the relevance that the optimization field has on each of the use cases inside PIACERE.

4.2.1 Modeled Optimization Problem

In a nutshell, the IOP is responsible for finding the best possible infrastructure given the input data received. This input data is provided in DOML format, and will include the functional requirements, non-functional requirements, the infrastructure model and the configuration. Then, the IOP performs the matchmaking for the infrastructure by the execution of an optimization intelligent technique using the FR and NFR against the available infrastructure and historical data, available from the catalogue of Infrastructural elements.

In other words, the optimizer will use an optimization algorithm, seeking for an optimized deployment configuration of the IaC on the appropriate infrastructural elements that best meet the predefined constraints (e.g., types of infrastructural elements, NFRs, and so on). The IOP will success if it is able to propose the most optimized deployment configuration of the infrastructural code taking into consideration the constraints predefined. To this end, several deployment configurations will be shown and ranked, each one with its main characteristics clearly accessible, and the user is the one which decides which solution is the chosen one.

In order to properly understand the problem modelled in the context of PIACERE, we introduce now a simplified example using a reduced catalogue of infrastructural elements. For doing this, we consider a simplified use case in which the user only needs to deploy three different elements: [`Virtual Machine, Database, Storage`].

Furthermore, for each of these elements, five different options are available:

- `Virtual Machine`: [A3_France, A2_USA, B8_Germany, C1_Spain, c2_Europe]
- `Database`: [dynamo.4, m3.medium, mysql.AZ_1, postgreSQL.G, r4.4xlarge]
- `Storage`: [Standard_Europe, Standard_USA, Storage3_Spain, G_1, AZ_3]

Each of these elements has some associated attributes such as cost, expected performance, availability, region, provider, type or capacity. With all this, the main problem is to find the best combination of [`Virtual Machine, Database, Storage`] which optimizes a certain objective or some defined objectives. In this specific example, we define the problem as a multi-objective one, seeking to optimize two different opposite objectives: the cost and the performance. Meanwhile, the problem can be identified as an integer programming problem, which is proved to be NP-Hard.

In order to find this best solution which optimizes both cost and performance objectives, the solving algorithms considered in this first stage of the project (which are detailed later) assign a numeric value to each option of every element. This way, an algorithm can define a tentative solution with this form:

```
[0, 3, 1]: [295.5, 125.2]
```

Considering the [`Virtual Machine, Database, Storage`] combination, this solution represents [A3_France, postgreSQL.G, Standard_USA], and `[295.5, 125.2]` is the cost associated to this solution (295.5) and its performance (125.2).

The feasible solution depicted above is just one of the many that can be found in the whole solution space. Being a NP-Hard problem, the algorithm is not capable of analysing all feasible solutions (in a real-world instance, bigger than the reduced example shown here). This way, intelligent algorithms build some feasible solutions and evolve them in a metaheuristic search process. This way, algorithms can find a set of solutions, called Pareto Front, that contains the best solutions explored for the problem at hand.

As a visual example, we show in the following Figure 3 some examples of Pareto fronts obtained by the prototype described in this document, using the preliminary data available in the current version of the infrastructural elements catalogue.

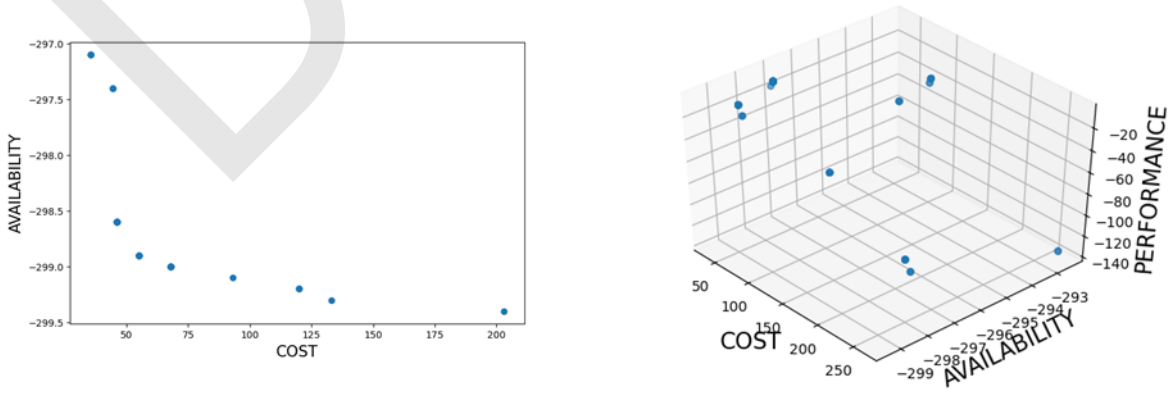

Figure 3: examples of Pareto front for the developed problem

4.2.2 Optimization relevance to the use-cases

SI-MPA Use-case

Assumption: we consider that the optimization of RAM, CPU, and DISK (on VMware) is done through mechanisms like: "balloon memory" for RAM (size) optimization, page sharing for CPU (cores availability) optimization, or thin provisioning for DISK space optimization. Additionally, there are also some settings available for "Resource Allocation Shares" which are priority settings for CPU-time access, for memory size access and for disk speed access.

To prevent "possible over-commits" from causing a disaster where some virtual machines configured together on a physical host, and considering that it could use more RAM, DISK or CPU than is physically available, some triggers can be configured to warn, move, freeze or shut down the most heavily used virtual machines to provide enough resources to other virtual machines. When virtual machines are configured within a cluster, a single virtual machine can be automatically live-migrated to another physical host (using vMotion) and/or to another physical LUN datastore (using SVMotion).

Next, these are the steps in this specific use-case:

- Step 1: Optimization of resources (CPU, RAM).
- Step 2: Optimization of the cost of resources.
- Step 3: Optimization of the price of licenses.

For doing these steps, we have two different options:

- Option 1: The desired infrastructure is installed in the canary environment. The stress tests shall be triggered. Resources vary so long that a minimum of resources is reached in pre-agreed response times.
- Option 2: The desired infrastructure is installed in a canary environment. An application is installed on the infrastructure. The stress tests shall be triggered. Resources vary so long as to achieve a minimum of resources in pre-agreed response times.

Despite these options are focused on the Step 1, both Option 1 and Option 2 cover also Step 2 and Step 3. Option 2 can cover the optimization of rental costs as well as license costs if the license costs are tied to the CPU or core. If the cost of licenses is tied to the number of users, then optimization at this level is not possible. The licensing policy should be evident from the catalogue. So, if the element in the catalogue is defined by additional parameters, such as the cost of the license on the CPU or core, then the optimization on the cost of the license can be performed otherwise not

Canary test pattern

In canary testing, we partially roll out a change and then we want to evaluate its performance against a baseline deployment. We want use A/B test pattern to test a hypothesis by using variant implementations. A/B is used to make government decisions based on the results derived from data. Our goal is to measure the effectiveness of functionality of virtual machine and docker nodes his new features and monitor any statistically significant difference in our new deployment.

Furthermore, we must consider complexity of our local environment, and we must ensure that deployment have zero impact on other production. For this reason, we have to be careful when we interact with third-party services.

5 Catalogue Implementation

5.1 Functional description

The purpose of the catalogue is to manage the different infrastructure elements using a data model that allows their classification based on dynamic properties that give flexibility to the inclusion of new elements. These characteristics associated with the services registered in the catalogue will allow user to choose an optimal deployment configuration by the optimization algorithms.

The dynamic information related to the cataloged service will be updated with the data provided by the monitoring component to be later included in the optimization algorithm and find the deployment that best suits the needs.

The catalogue will also provide status information for the deployed instances and receive information about tore down instances.

5.2 Requirements covered by this prototype

The user's requirements satisfied by this final version are described in Table 3. All these requirements will be further polished and adapted in future stages of the project. For this reason, they are adapted to this first stage of the project. They are aligned with D2.1 deliverable, which details the requirements.

Table 3: Requirements covered

Req ID	Description	Status
WP5.3-REQ3	IOP will include a catalogue of infrastructural elements (e.g. (edge, node) computation, networks, cloud services (e.g. IaaS, PaaS, SaaS) classifiable by a set of constraints (e.g. memory, disk, …). This catalogue of infrastructural elements should be clearly defined, including possible restrictions and dynamic variations. These infrastructural elements will be transformed as optimization variables, and they will be intelligently treated by the optimization algorithm seeking to find the best configuration deployment.	Partially satisfied
WP5.3-REQ4	Provide the means for the IOP to properly consume all the data related to the catalogue of infrastructural elements status, as well as their characteristics and possible variations. Special mention shall be done here to the values monitored by the self-learning algorithm / monitoring component. This module shall provide real measures regarding the infrastructural elements in order to update their characteristics.	Satisfied

5.2.1 Fitting into overall PIACERE Architecture

The Infrastructural Elements Catalogue (IEC) is one of the components of the PIACERE architecture. It interacts with other tools in the PIACERE ecosystem:

- IDE Developer includes element information in the Catalogue.
- IOP, provides information to the IOP about available elements and its related dynamic information.
- Runtime Controller receives information from the IEM (through the Runtime Controller) about deployed instances.

- Runtime Controller, receives monitoring information from the Runtime Monitoring (through the Runtime Controller)
- Runtime Controller receives information about tore down instances.

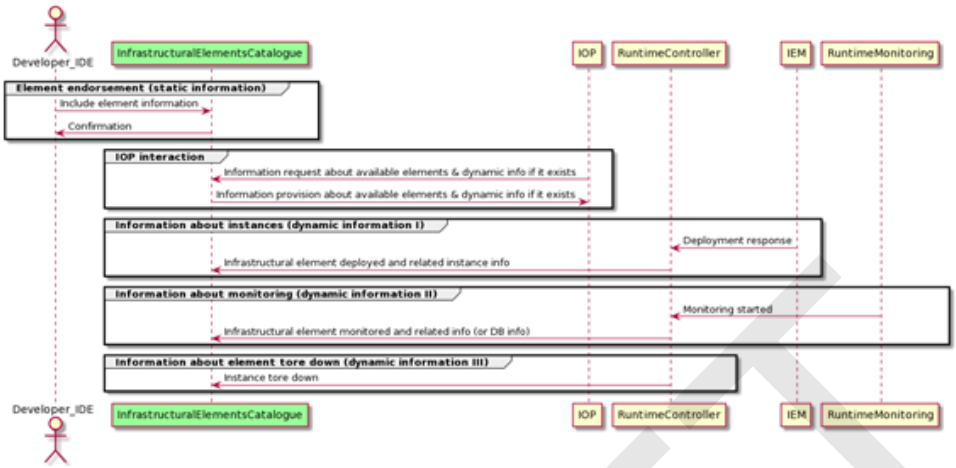

Figure 4: Sequence Diagram of the Infrastructural Elements Catalogue

5.3 Technical description

This subsection is devoted to describing the technical specification of this first prototype. First, the main architecture of the prototype is shown and described in Section 5.*3.1. Prototype Architecture*. After that, all components of the prototype are detailed in Section 5.*3.2 Components Description*. This subsection finishes with the technical specifications of the developed system in *Section 5.3.3. Technical Specifications*.

5.3.1 Prototype architecture

IEC architecture is based in a microservices style which splits the front-end and the backend, so that´s it´s easier to scale and survive infrastructure issues.

The main purposes of these components are described in the following Section 5.*3.2 Components Description*.

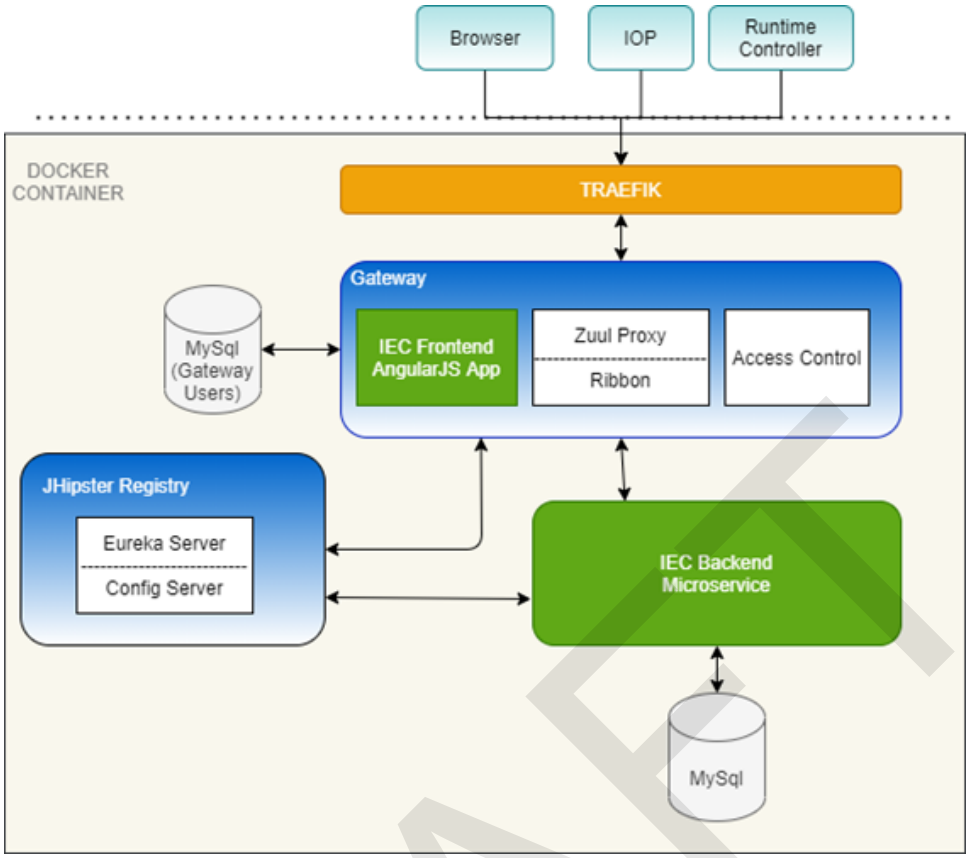

Figure 5: Main architecture of the Infrastructural Elements Catalogue

5.3.2 Component's description

This first prototype of the IEC is composed by two principal components, which main purposes are briefly described as follows. It should be noted that further detail of this components is provided in the upcoming *Section 7.1 Package information*.

- `IEC-Frontend`: this first component is the entry point of the IEC system. All the external callings to the whole IEC should be made through this initial component. There are two different uses of this component, the frontend for managing the service Catalogue and the API exposed services to interact with other components.
- `IEC-Backend`: this component cannot be accessed externally. It manages all the logic of the IEC and exposes the rest API services for the related components.

In addition, some considerations about the other components of the IEC infrastructure:

- Access control. JSON Web Token (JWT) mechanism used. A stateless security mechanism which uses a secure token that holds the user´s login name and authorities.
- Data persistence in MySQL database.
- JHipster Registry. Service discovery using Netflix Eureka.

5.3.3 Technical specifications

We show in this section the technical specification of the prototype:

- This prototype has been developed using JHipster Framework.

- The framework provides all the needed for a modern web application and microservice architecture.
- It uses Spring boot for application configuration.
- In the client side, IEC-Frontend gateway uses Yeoman, Webpack, Angular and Bootstrap technologies.
- In the server side, IEC-Backend microservice, uses Maven, Spring, Spring MVC Rest, Spring Data JPA and Netflix OSS.

6 IOP Implementation

6.1 Functional description

The optimization problem formulated in PIACERE and solved by the IOP consists of having a service to be deployed and a catalogue of infrastructural elements, with the principal challenge of finding an optimized deployment configuration of the IaC on the appropriate infrastructural elements that best meet the predefined constraints (e.g., types of infrastructural elements, NFRs, and so on). In this context, the IOP should be able to propose the most optimized deployment configuration of the infrastructural code taking into consideration the constraints predefined. To this end, several deployment configurations will be shown and ranked.

With all this, this first implementation of the IOP consists of a laboratory prototype which works with a reduced, controlled, and static Infrastructural Elements Catalogue. Being a first prototype, it includes a wide variety of multi-objective algorithms, and two different optimization formulations (continuous and discrete). Furthermore, in order to be able to decide which algorithm will be used and under which framework, two different well-known optimization frameworks are considered in this system: jMetal and MOEA. In overall, 40 different algorithm approaches have been considered and compared in this prototype, which arguably help us to define the final optimization algorithm for the whole PIACERE system.

The main innovation of this prototype is mainly the formulation of the whole optimization problem, which can optimize different objectives depending on the user's needs. At this moment, three different objectives have been considered, which can be faced simultaneously: cost, availability, and performance. In future stages of this prototype, these objectives, as well as the optimization constraints and user requirements, will be obtained from the DOML.

6.2 Requirements covered by this prototype

The user requirements satisfied by this interim version are described in Table 4. All these requirements have been obtained from the *PIACERE WP2 Requirements* internal document.

Table 4: user requirements satisfied by the optimizer.

Req ID	Description	Status	Requirement Coverage
REQ02	The objectives that will help to the formulation of the optimization problem shall be defined. Usually, the objective function of an optimization problem is purely built using functional requirements. In any case, non-functional objectives through the optimization perspective shall also be considered for studying its inclusion in the objective function	Discarded	This requirement was finally discarded.
REQ05	The outcomes of the IOP shall be defined, in terms of content and format, in order to fully and easily integrate with other work packages and tasks.	Discarded	This requirement was finally discarded.
REQ06	The optimization algorithm shall constantly use the Canary Environment for different purposes that can range from testing along	Discarded	This requirement was finally discarded.

Req ID	Description	Status	Requirement Coverage
	the search process to verification of the provided solutions. For this reason, and because of the frequent use of this resource, the IoP shall have a fast and easy access to the Canary Environment, which shall be defined.		
REQ98	The IOP components provide feedback on the DOML code, without doing automatic writes. The end user can choose to accept or not the feedback received.	In progress	At this stage of the project, we are still defining the whole structure of the DOML, for this reason, the first prototype of the IOP has not considered this language.

The internal requirements satisfied by this interim version are described in Table 5. All these requirements are as well polished and adapted as the project advances. All these requirements will be further polished and adapted in future stages of the project. For this reason, they are adapted to this first stage of the project.

Table 5: internal requirements satisfied by the optimizer

Req ID	Description	Status	Requirement Coverage
WP5.3-REQ1	Load/read information about the catalogue of infrastructural elements	Partially satisfied	A preliminary catalogue of infrastructural elements is successfully loaded and read by the algorithm.
WP5.3-REQ2	IOP shall consider both functional and non-functional requirements for searching the best solutions that fit the user demand.	Satisfied	The IOP prototype considers these requirements to build the fitness function of each solver algorithm. These requirements will be used to define each of the objectives that algorithms should optimize.
WP5.3-REQ3	IOP shall use optimization algorithms such as genetic algorithms and provide a set of potential combinations of elements that fulfill the established user requirements.	Satisfied	Algorithms such as NSGA-II, MOCell, SMPSO or SPEA have been considered in this prototype. These algorithm search through different combinations of elements for the services and find the optimal combination of them.

6.2.1 Fitting into overall PIACERE Architecture

The IOP is one of the components of the PIACERE architecture. It is present mainly in the phase of PIACERE Run Time (Figure 6). The IOP interacts with other tools in the PIACERE ecosystem. To this respect:

- IOP receives information from the Infrastructural Elements Catalogue regarding the elements that will have to be used for optimizing the functional and non-functional requirements and will return the optimized solution to the Runtime Controller.
- The Runtime Controller communicates with the IOP for obtaining the best configuration for deploying a certain service. For optimizing the problem, the IOP will gather the optimization objectives and optimization requirements from the DOML, which is also provided by the Runtime Controller.

- Self-Healing mechanism warns IOP when any CPU idle exceeds a threshold and asks for a dynamic optimization in case this performance downturn requires it.

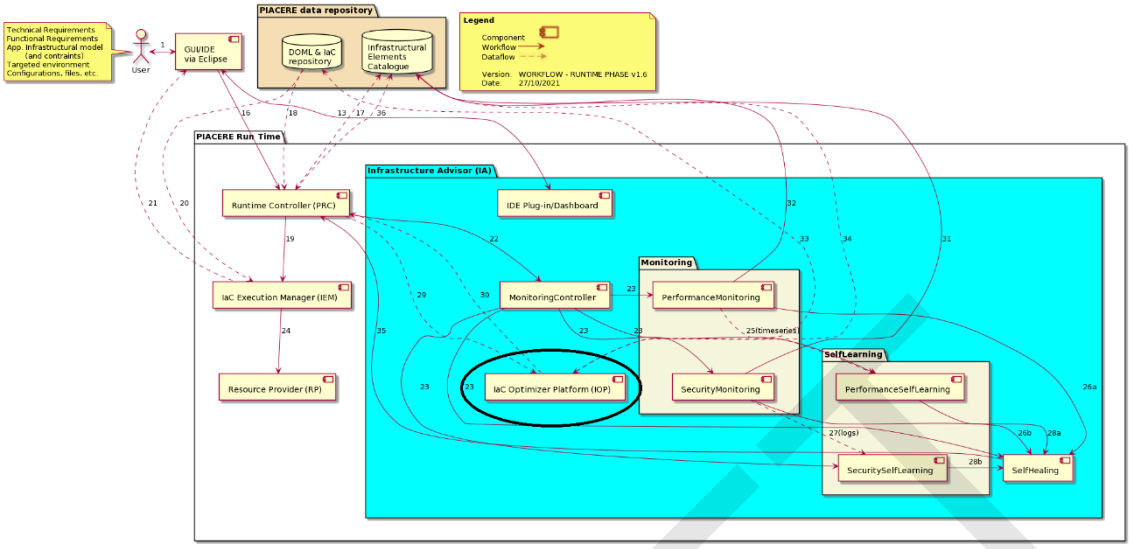

Figure 6: PIACERE Runtime Diagram on its 1.6 version

6.3 Technical description

This subsection is devoted to describing the technical specification of this first prototype. First, the main architecture of the prototype is shown and described in Section *6.3.1. Prototype Architecture*. After that, all components of the prototype are detailed in Section *6.3.2 Components Description*. This subsection finishes with the technical specifications of the developed system in *Section 6.3.3. Technical Specifications*.

6.3.1 Justification of the optimization frameworks used

In the literature, some interesting multi-objective optimization framework can be identified, which has been potentially usable for the development of this IOP prototype. In the context of PIACERE, a comprehensive research has been made to determine which of these platforms are more adequate for being finally considered.

As a result of this research process, two are the frameworks that have been finally considered for the implementation of the prototype: jMetal[1] and MOEA. On the one hand, jMetal is one of the most used optimization frameworks for solving multi-objective optimization problems. It is an open-source platform which counts with a remarkable community of researchers supporting it and developing new functionalities in a frequent way. It permits the adoption of several optimization types, such as continuous, binary, real or dynamic, and it has been extensively employed by the optimization community for solving both academic and real-world problems. Finally, it is fully adaptable to PIACERE problems and it counts with more than 20 different multi-objective algorithms for being used. For all these reasons, we have deemed it as one of the platforms employed in our prototype.

On the other hand, MOEA is another reputed framework for dealing with multi-objective optimization problems. As can be seen in the webpage of the platform *MOEA*[2] framework *"is a free and open source Java library for developing and experimenting with multi-objective evolutionary algorithms and other general-purpose single and multi-objective optimization*

[1] jmetal.github.io/JMetal/
[2] moeaframework.org

algorithms". Its open source nature makes easy not only the adaptation of the algorithms to the PIACERE problem, but also their extension to add custom mechanisms and functionalities. MOEA permits the usage of more than 20 algorithms, making the framework one of the most complete of the literature. Furthermore, the adaptation of the algorithms to any optimization problem can be made in an easy way, as well as their full configuration. All these arguments, the existence of a quite large community using it and the availability of a valuable documentation has led us to consider it in the context of this project.

Thus, the consideration of jMetal and MOEA allows us the adaptation of a significant amount of methods, both classical (as the NSGA-II) and sophisticated ones (as the WASFGA). Also, we can count on algorithms from different research streams such as Swarm Intelligence (SMSPSO, OMOPSO), Evolutionary Computation (MOCell, NSGA-II, MOEA, etc.). For this reason, and because these libraries are the most reputed ones in the literature for being used in JAVA, we have considered them to conduct ambitious and meaningful experimentations.

In any case, for reaching this technological conclusion, additional research has been made, which is worthy to be mentioned here. In overall, we have studied the adoption of three additional frameworks, which have been finally discarded to be considered in the prototype: ECJ[3], JCLEC-MO[4] and Jenetics[5]. The first of these libraries, ECJ, has been discarded because of its clear trend to a single-objective optimization. For this reason, the multi-objective algorithms available are not only a few, but also the most classical ones, such as NSGA-II, NSGA-III and PSO. Thus, the adoption of ECJ would not add value to the prototype.

The second of the discarded platforms has been JCLEC-MO. Despite this framework providing a significant number of multi-objective algorithms (up to 15), it has the principal problem of not being much optimized for being externally used. This fact makes it difficult to configure each of the available solvers. Additionally, lots of dependencies are needed to efficiently run JCLEC-MO, i.e. JCLEC base, datapro4j library core, Apache Commons libraries, JUnit and harmcrest-core. Furthermore, the binary jar library is not available, making difficult the efficient import to PIACERE prototype.

Lastly, despite Jenetics is an easily accessible framework thanks to Maven, and that it has lot of potential, it is generalist library for evolutionary computation. For this reason, and despite it includes functionalities for multi-objective optimization, it does not include algorithms for being adapted to the PIACERE problem. This is the main reason because it does not suppose any advantage in comparison to MOEA and jMetal.

6.3.2 Prototype architecture

The main architecture of this first IOP prototype is depicted in the Figure 7. In this architecture, four different components can be distinguished: `Algorithm-runner`, `Data Reader`, `Algorithm Frameworks` and `Solution Processing Mechanism`. The main purposes of these components are described in the following Section *6.3.2 Components Description*.

[3] https://cs.gmu.edu/~eclab/projects/ecj/
[4] https://www.uco.es/kdis/research/software/jclec-mo/
[5] https://jenetics.io/

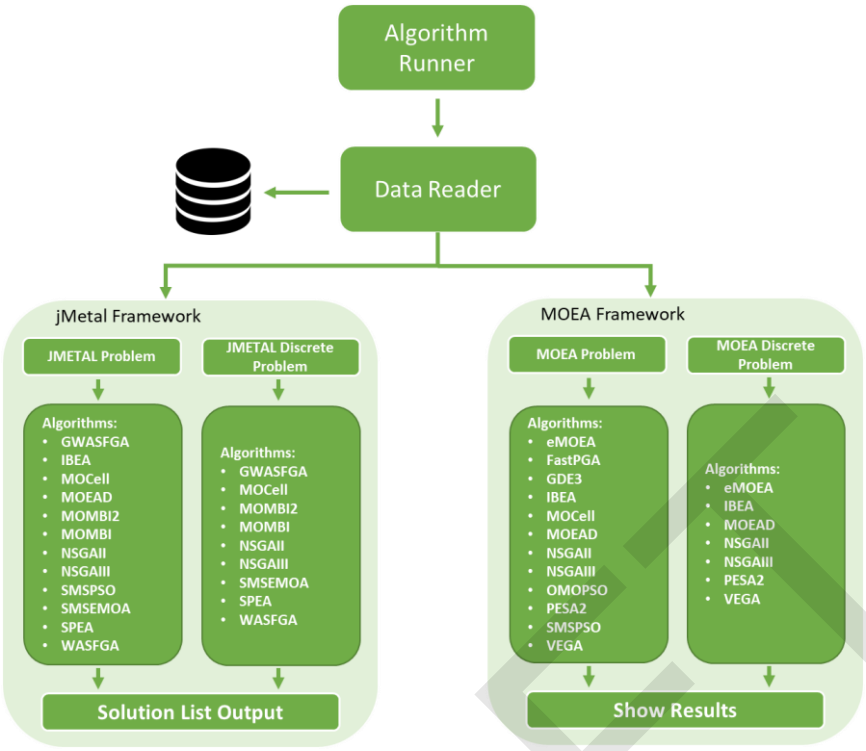

Figure 7: IOP first prototype architecture

6.3.3 Components description

This first prototype of the IOP optimization system is composed by four different components, which main purpose are briefly described as follows. It should be noted that further detail of this components is provided in the upcoming *Section 8.1 Package information*.

- `Algorithm-runner`: this first component is the entry point of the IOP optimization system. All the external callings to the whole IOP should be made through this initial component. In a nutshell, the algorithm runner is in charge of gathering all the input data (which in the future will be provided by the DOML), prepare all the elements that will be used in the whole procedure, initialize optimization algorithms and both data reader and solving algorithms.
- `Data Reader`: this component is devoted to gathering the data from the Infrastructural Elements Catalogue and process them in order to be used by the solving algorithms. To do this, this component should conduct some procedures such as data cleaning, normalization, discretization, or integration.
- `Algorithm Frameworks`: This component is the one in which solving algorithms are present. At this stage of the prototype, two different frameworks are available: jMetal and MOEA. For each of this platforms, different approaches are already adapted for solving the PIACERE problem described in this deliverable. Furthermore, two different types of algorithms are available, which are characterized by the kind of variables they use: continuous and discrete.
- `Solution Processing Mechanism`: the main purpose of this component is to gather the results obtained by each algorithm of both frameworks and return them to the user in an understandable form. It should be noted that for each framework considered, a different mechanism is needed. Finally, in future stages, this component will be the one in charge of translating the solution to DOML format.

6.3.4 Technical specifications

This prototype has been developed using JAVA, which is a class-based, high level, object-oriented and general-purpose programming language. It is mainly conceived for having as few dependencies as possible, making it more efficient in this aspect. JAVA follows the known as WORA philosophy, which is described as *Write Once and Run Anywhere*. This means that compiled JAVA code can be run in all platforms that support JAVA without requiring any additional recompilation.

Furthermore, jMetal and MOEA frameworks are imported using the MAVEN mechanism, which is a software project management and comprehension tool. MAVEN is inspired by a concept coined as Project Object Model (POM). Thus, MAVEN can manage the building, reporting and documentation in a project from a unique file of information. Regarding the versions employed, for jMetal version 5.1 has been employed, for MOEA 2.13 and for JAVA the 11 JDK.

6.4 Connection among IOP prototype and the Catalogue of Infrastructural Elements

One of the most important aspects in the development of the whole T5.3 is the correct connection between the two main components implemented in this task: the IOP and the Catalogue of Infrastructural Elements. In this subsection, we describe how this connection has been carried out in this first prototype of the IOP.

Taking as the basis for the description of the catalogue, a specific method has been built in the dedicated API, which main objective is to obtain the whole catalogue of infrastructural elements. We can find this method in the *iecbackend* of the API, in the category *root-service-resource*, and with the name */api/root-services/catalogue*.

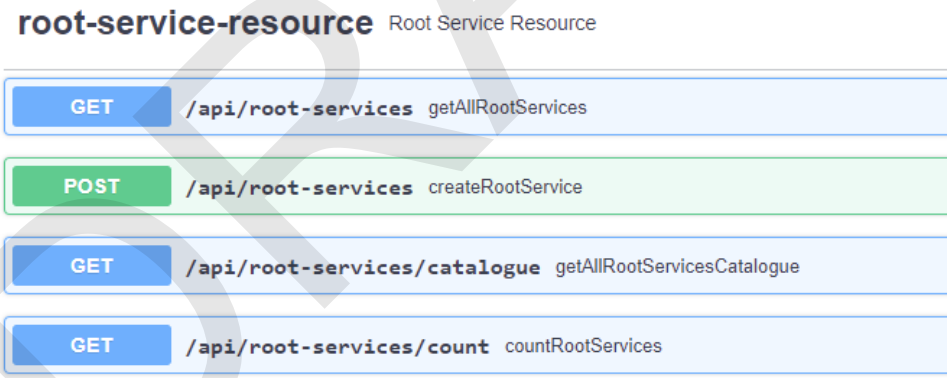

Figure 8: screenshot of the catalogue showing the method for obtaining the whole data

Thus, for properly using this method in the IOP prototype, we have added the following URL in the class `data_reader`, which is in charge of accessing the catalogue, and obtaining and processing the information that contains, as will be explained later. This is the URL employed:

```
URL url = new URL("https://iec-frontend.piacere.esilab.org:8444/services/iecbackend/api/root-services/catalogue");
```

Figure 9: URL used for accessing the catalogue of infrastructural elements

Coming back to the API developed for the catalogue, and after launching an execution example of the method, we can see that the requested information needed to properly run the method externaly is the following ones:

Curl

```
curl -X 'GET' \
  'https://iec-frontend.piacere.esilab.org:8444/services/iecbackend/api/root-services/catalogue' \
  -H 'accept: */*' \
  -H 'Authorization: Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJhZG1pbiIsImF1dGgiOiJST0xFX0FETUlOLFJPTEVfVVN
```

Figure 10: Request instructions needed for obtaining the information of the catalogue

Thus, in order to properly call the method with all the information needed, the JAVA class `data_reader` has been completed using the following code:

```
// Create a trust manager that does not validate certificate chains
TrustManager[] trustAllCerts = new TrustManager[]{new X509TrustManager(){
    public X509Certificate[] getAcceptedIssuers(){return null;}
    public void checkClientTrusted(X509Certificate[] certs, String authType){}
    public void checkServerTrusted(X509Certificate[] certs, String authType){}
}};

// Install the all-trusting trust manager

SSLContext sc = SSLContext.getInstance("TLS");
sc.init(null, trustAllCerts, new SecureRandom());
HttpsURLConnection.setDefaultSSLSocketFactory(sc.getSocketFactory());

HttpURLConnection http = (HttpURLConnection)url.openConnection();
http.setRequestProperty("Accept", "application/json");
http.setRequestProperty("Authorization", "Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJhZG1pbiIsImF1dGgiOiJST0xFX0"
        + "FETUlOLFJPTEVfVVNFUiIsImV4cCI6MTYzMTk1MjYyM30.dZ_-D0bf6RopIhFQZY36M1HNZACOd398N89ljU4x2Hqw-cOY3AwV"
        + "urLQQCq6N9g0MS0cttUAJoMgtsCXvyy_Cg");
```

Figure 11: Steps followed for properly calling the API method from the IOP prototype

As it can be seen, the main step at this point is to create a `HttpURLConnection` entity, initializing it with the URL shown in Figure 11. After that, and before calling the method, all the request information is introduced using the `.setRequestProperty` method. Once conducted these steps, the method is called, and in case that the response code obtained is the correct one, we extract the information obtained from an external JSON file, which is later processed for extracting all the needed information for conducting the optimization procedure. The method used for obtaining the code is `.getResponseCode()`, while the one employed for obtaining the information returned is `.getResponseMessage()`. We show in Figure 12 this last process.

```
System.out.println(http.getResponseCode() + " " + http.getResponseMessage());
BufferedReader br = null;
if (http.getResponseCode() == 200) {
    br = new BufferedReader(new InputStreamReader(http.getInputStream()));
    String strCurrentLine;
        while ((strCurrentLine = br.readLine()) != null) {
            PrintWriter writer = new PrintWriter("data_catalogue.json", "UTF-8");
            writer.println(strCurrentLine);
            writer.close();
        }
} else {
    br = new BufferedReader(new InputStreamReader(http.getErrorStream()));
    String strCurrentLine;
        while ((strCurrentLine = br.readLine()) != null) {
                System.out.println(strCurrentLine);
        }
}
http.disconnect();
```

Figure 12: JAVA code used for obtaining the information returned from the API call.

Finally, and for the sake of completeness, we depict in Figure 13 a brief excerpt of the JSON generated with the information obtained from the API.


```
{
   "id":1,
   "serviceName":"Cl_Spain",
   "deletedDate":null,
   "serviceClass":{
      "id":3,
      "serviceClassName":"Virtual Machine",
      "deletedDate":null
   },
   "serviceAttributeValues":[
      {
         "id":1,
         "serviceAttributeValue":"00EU",
         "unitValue":"",
         "serviceAttributeType":{
            "id":1,
            "name":"Region",
            "nfrName":"Region",
            "isEnumeration":true,
            "isForm":false,
            "isCommon":true,
            "isFunctionalRequirement":false,
            "units":"",
            "unitFactor":"",
            "unitRule":"LIKE",
            "evalRule":""
         }
      },
      {
         "id":2,
         "serviceAttributeValue":"SPEU",
         "unitValue":"",
         "serviceAttributeType":{
            "id":2,
            "name":"Zone",
            "nfrName":"Zone",
            "isEnumeration":true,
            "isForm":false,
            "isCommon":true,
            "isFunctionalRequirement":false,
            "units":"",
            "unitFactor":"",
            "unitRule":"LIKE",
            "evalRule":""
```


Figure 13: Excerpt of information retrieved from the API built for Catalogue of Infrastructural Elements.

7 Catalogue Delivery and usage

7.1 Package information

7.1.1 IEC-Backend

The main structure of the prototype developed in this first stage of the project is composed by the packages shown in the following Figure 13.

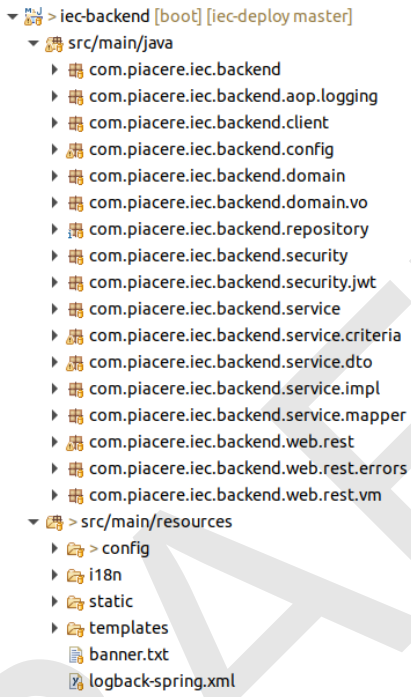

Figure 14: Structure of the catalogue

Each of these packages has its main objective and its context within the whole prototype. Furthermore, these packages are also comprised by several JAVA classes. With all this, the main purpose and composition of each component is as follows:

- `com.piacere.iec.backend.aop.logging`: this package is composed of `LoggingAspect.java` which define the aspect for logging execution of service and repository Spring components.
- `com.piacere.iec.backend.aop.client`: Composed by `UserFeignClientInterceptor.java` which implements `RequestInterceptor.java`. This class checks and add JWT token to the request header.
- `com.piacere.iec.backend.aop.config`: this package contains all classes related to configuration purposes.

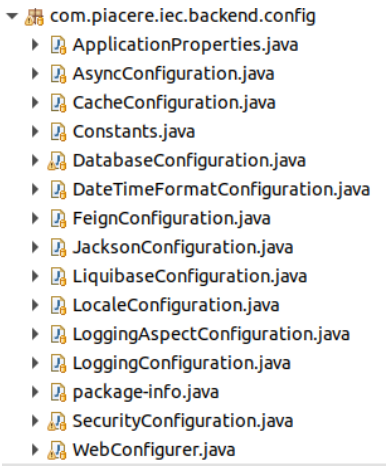

Figure 15: structure of config package

- `com.piacere.iec.backend.aop.domain`: this package contains data model classes.

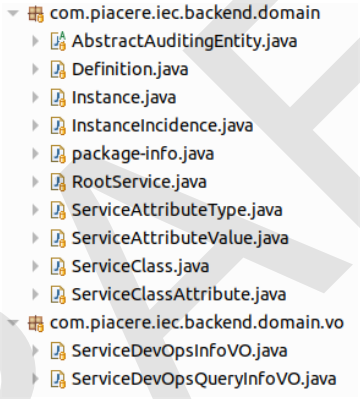

Figure 16: structure of domain package

- `com.piacere.iec.backend.repository`: this package contains Spring Data SQL repository classes.
- `com.piacere.iec.backend.security`: this package contains Spring Security related classes for security management.
- `com.piacere.iec.backend.service`: this package contains Iec-backend services for CRUD operations and other requirements needed.
- `com.piacere.iec.backend.web`: this package contains classes to expose Iec-backend rest end points.

7.1.2 IEC-Frontend

The main structure of the prototype developed in this first stage of the project is composed of JAVA classes related to Spring boot project and AngularJS files.

7.1.2.1 Spring boot package information

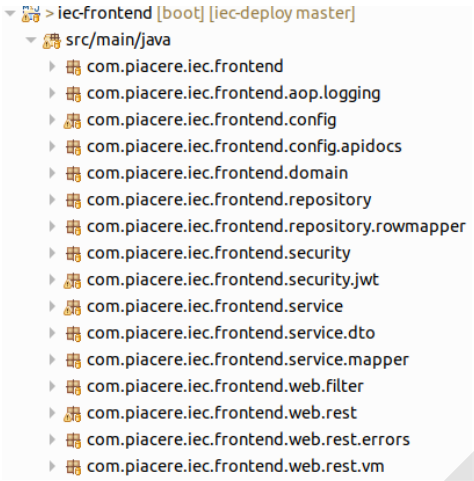

Figure 17: relation of packages

Each of these packages has its main objective and its context within the whole prototype. Furthermore, these packages are also comprised by several JAVA classes. With all this, the main purpose and composition of each component is as follows:

- `com.piacere.iec.frontend.aop.logging`: This package is composed by `LoggingAspect.java` which define the aspect for logging execution of service and repository Spring components.
- `com.piacere.iec.frontend.config`: This package contains all classes related to configuration purposes.
- `com.piacere.iec.frontend.domain`: This package contains user data model and Authority classes.
- `com.piacere.iec.frontend.repository`: This package contains Spring Data SQL repository classes for user and security management.
- `com.piacere.iec.frontend.security`: This package contains Spring Security related classes for security management.
- `com.piacere.iec.frontend.service`: This package contains Iec-frontend services for CRUD operations and other requirements needed for user and security management.
- `com.piacere.iec.frontend.web`: This package contains classes to expose Iec-frontend rest end points for user and security management.

7.1.2.2 AngularJS package information

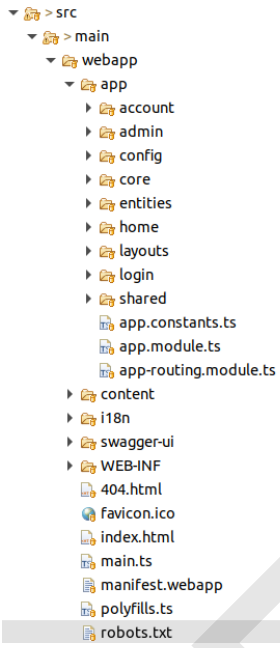

Figure 18: AngularJS package

Each of these packages and typescript files has its main objective and its context within the whole prototype.

The main purpose and composition of each file/component is as follows:

- `app/app.constants.ts`: Global application constants.
- `app/app.module.ts`: Declaration of all the needed modules, providers and components loaded in the web application.
- `app/app-routing.module.ts`: Routing configuration.
- `app/account`: Account management module. It contains components and services related to the user account management.
- `app/admin`: Admin related modules. Api documentation module, Gateway route module and user management module.
- `app/config`: Global configuration and constant typescript files.
- `app/core`: Global utils files, core models and services.
- `app/entities/iecBackend`: All files related to the iecBackend data model. Services, components and models.
- `app/home`: Home component
- `app/layout`: Layout components; error, footer, navbar, main and profile components.
- `app/login`: User Login component.
- `app/shared`: Shared module with common components, directives and pipes.
- `content`: Static files of webapp. Css and images.
- `i18n`: Internationalization files.

7.2 Installation instructions

This project is executed in a docker container.

There are docker compose files for each environment development/production.

To execute this project in a development environment:

- *docker-compose --env-file .env.dev -f docker-compose-local-dev.yaml up --build -d*Frontends

Available services after initialization:

- JHipster registry: http://locahost:8761
- IecFrontend web app: http://localhost:8080
- iecBackEnd Api Documentation: http://localhost:8080/services/iecbackend/v3/api-docs

7.3 User Manual

In this subsection we will detail the five steps that must be done in order to deploy and test this prototype of the Infrastructural Service Catalogue.

1. Clone repository
   git clone https://git.code.tecnalia.com/piacere/private/t53-iop-catalogue/iec-deploy.git

2. Run docker compose to start Jhipster registry and MySQL instances
   docker-compose --env-file .env.dev -f docker-compose-local-dev.yaml up --build -d

3. Build and deploy Catalogue backend
   ./mvnw -Pdev,api-docs -DskipTests

4. Build and deploy Catalogue frontend
   ./mvnw -Pdev,webapp,api-docs -DskipTests

5. Access Piacere Catalogue web application
   http://localhost:8080

   ***User with admin role (user/password)**: admin / admin*
   ***Non admin role user (user/password)**: user / user*

7.4 Licensing information

Information about license not included yet.

7.5 Download

The code is available at Tecnalia GitLab repository:

https://git.code.tecnalia.com/piacere/private/t53-iop-catalogue/iec-deploy

The source code will be available on the public git repository and accessible through the project's website https://www.piacere-project.eu/. At the time of writing this deliverable, the source code is provided under request through an email to the address appearing on the website (https://www.piacere-project.eu/) in the footer under "Contact Us".

8 IOP Delivery and usage

8.1 Package information

The main structure of the prototype developed in this first stage of the project is composed of the packages shown in the following Figure 19.

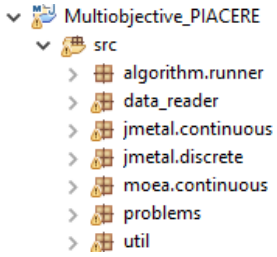

Figure 19: Main structure of the developed prototype.

Each of these packages has its main objective and its own context within the whole prototype. Furthermore, these packages are also comprised by several JAVA classes. With all this, the main purpose and composition of each component is as follows:

- `algorithm.runner`: this package is composed by two different classes `JMETALAlgorithmRunner.java` and `MOEAAlgorithmRunner.java` which are the main entry points of the system, and which are in charge of launching the whole experimentation for both jMetal and MOEA Framework.
- `data_reader`: this component is also comprised by one sole class named Rile_Reader.java, which main objective is to access the Catalogue of Infrastructural Elements and extract all the information needed to conduct the optimization. Once extracted, this data is formatted to be legible by the optimization algorithms.
- `jmetal.continuous`: this package contains some of the algorithms considered from the jMetal framework on this first prototype of PIACERE IOP. Furthermore, approaches within this package are characterized by defining the optimization variables as `double` values. At this stage, 12 different solvers of this kind have been adapted to solving the problem modelled in this prototype. This package is built as follows:

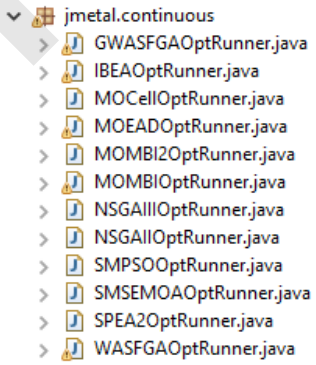

Figure 20: composition of `jmetal.continuous` package.

- `jmetal.discrete`: this package contains the rest of the algorithms considered from the jMetal Framework on this first prototype of PIACERE IOP. Furthermore, approaches within this package are characterized by defining the optimization variables as `integer` values. At this stage, 12 different solvers of this kind have been adapted to solving the problem modelled in this prototype. This package is built as follows:

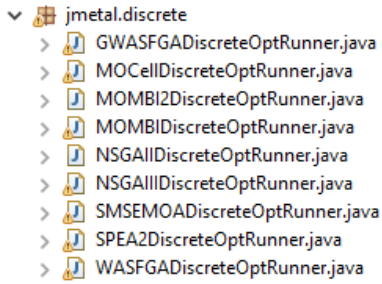

Figure 21: composition of `jmetal.discrete` package.

- `moea.algorithm`: this package contains all the algorithms embraced from the MOEA framework for this first prototype of PIACERE IOP. Differently from jMetal, approaches within this package are characterized by being able of solving both `continuous` and `discrete` problems. At this stage, 12 different solvers of this kind have been adapted to solving the problem modelled in this prototype. This package is built as follows.

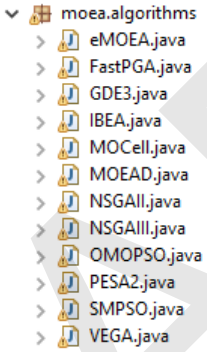

Figure 22: composition of `moea.algorithms` package

- `problems`: this package is composed by six different classes. These classes are the ones in charge of modelling the optimization problem, specifying the optimizing variables of the problem, their associated constraints, defining the objective function, and calculating the value of each of the considered optimizing objectives. On the one hand, the first four classes of this package are related with the definition of continuous and discrete formulations on jMetal: `JMETAL_Problem`, `JMETAL_ProblemDiscrete`, `JMETAL_ProblemJM` and `JMETAL_ProblemDiscreteJMDiscrete`. On the other hand, the last two classes are focused on formulating the problem for being solved using the MOEA framework: `MOEA_Problem` and `MOEA_ProblemDiscrete`. These are the main classes that compose this package.

Figure 23: structure of the package `problems`.

- `util`: this last package is comprised by two different classes `ShowResults.java` and `SolutionListOutput.java`, which main purpose is to gather the results obtained by each algorithm of both frameworks and return them to the user in an

understandable form. In future stages. In the future, these will be the classes in charge of traducing the solution to DOML format.

8.2 Installation instructions

The prototype is provided in a compressed folder, which should be imported from any JAVA development framework, such as Eclipse or NetBeans. Besides, it should be also considered that jMetal and MOEA frameworks are imported in the prototype using Maven functionality.

Furthermore, the jMetal version employed in this prototype is the one that corresponds with the last release (at the time of writing this deliverable), which is the 5.1 version. The same can be said about MOEA framework, which last version is the 2.13. With all this, the main requirements of the prototype are: MAVEN, JAVA 11 JDK and a JAVA IDE platform (such as Eclipse). Finally, we depict here the dependencies added to the `pom.xml` file, in order to properly import both jMetal and MOEA frameworks to the whole project.

```
<dependency>
    <groupId>org.uma.jmetal</groupId>
    <artifactId>jmetal-core</artifactId>
    <version>5.1</version>
</dependency>

<dependency>
     <groupId>org.uma.jmetal</groupId>
     <artifactId>jmetal-problem</artifactId>
     <version>5.1</version>
</dependency>

<dependency>
    <groupId>org.uma.jmetal</groupId>
    <artifactId>jmetal-algorithm</artifactId>
    <version>5.1</version>
</dependency>

<dependency>
    <groupId>org.uma.jmetal</groupId>
    <artifactId>jmetal-exec</artifactId>
    <version>5.1</version>
</dependency>

<dependency>
     <groupId>org.moeaframework</groupId>
     <artifactId>moeaframework</artifactId>
     <version>2.13</version>
</dependency>
```

8.3 User Manual

The entry points of the prototype are the classes coined as `JMETALAlgorithmRunner.java` and `MOEAAlgorithmRunner.java`, which are within the package `algorithm.runner`. In these classes, all the execution commands can be found. A user can add or remove any of these callings to conduct an experimentation adapted to its requisites. On the one hand, we depict in Figure 24 an excerpt of the class `JMETALAlgorithmRunner.java`, devoted to jMetal framework.

```
Problem p = new Problem("data/CatalogueServices.json");

System.out.println("Running NSGA-II Opt") ;
new NSGAIIOptRunner(p).run();

System.out.println("Running MOCell Opt") ;
new MOCellOptRunner(p).run();

System.out.println("Running SMPSO Opt") ;
new SMPSOOptRunner(p).run();

ProblemDiscrete pd = new ProblemDiscrete("data/CatalogueServices.json");

System.out.println("Running Discrete NSGA-II Opt") ;
new NSGAIIDiscreteOptRunner(pd).run();

System.out.println("Running Discrete NSGA-III Opt") ;
new NSGAIIIDiscreteOptRunner(pd).run();

System.out.println("Running Discrete MOMBI Opt") ;
new MOMBIDiscreteOptRunner(pd).run();
```

Figure 24: An excerpt of the jMETALAlgorithmRunner.java class.

On the other hand, we show in Figure 25 an excerpt of the launching class `MOEAAlgorithmRunner.java`, which is focused on running the experimentation regarding MOEA Framework.

```
MOEA_Problem problem = new MOEA_Problem(keySet.size());
Data_Reader reader = new Data_Reader("data/CatalogueServices.json");
MOEA_Problem.mapOfElements = reader.mapOfElements;

NondominatedPopulation result;

result = GDE3.GDE3_runner(problem);
ShowResults.showResults(result,"GDE3",result.get(0).getNumberOfObjectives());

result = eMOEA.eMOEA_runner(problem);
ShowResults.showResults(result,"eMOEA",result.get(0).getNumberOfObjectives());

result = NSGAII.NSGAII_runner(problem);
ShowResults.showResults(result,"NSGAII",result.get(0).getNumberOfObjectives());

MOEA_ProblemDiscrete problemDiscrete = new MOEA_ProblemDiscrete(keySet.size());
MOEA_ProblemDiscrete.mapOfElements = reader.mapOfElements;

result = eMOEA.eMOEA_runner(problemDiscrete);
ShowResults.showResults(result,"eMOEA_Disc",result.get(0).getNumberOfObjectives());

result = NSGAII.NSGAII_runner(problemDiscrete);
ShowResults.showResults(result,"NSGAII_Disc",result.get(0).getNumberOfObjectives());

result = NSGAIII.NSGAIII_runner(problemDiscrete);
ShowResults.showResults(result,"NSGAIII_Disc",result.get(0).getNumberOfObjectives());
```

Figure 25: An excerpt of the MOEAAlgorithmRunner.java class.

Furthermore, as mentioned in previous sections, the developed prototype can contemplate the resolution of two or three objectives. This election of optimizing objectives is a configurable aspect, which must be contemplated in the code. More concretely, the classes `JMETAL_ProblemJM.java` and `JMETAL_ProblemJMDiscrete.java` in package `problems` are the one that should be modified in order to consider all this issue for the case of jMetal framework. In these classes, the following line can be found:

```
this.setNumberOfObjectives(X);
```

In which X should take a value equal to 2 or 3, depending on the optimizing objectives. Furthermore, we can find in the same class the following lines, which are the ones that configure the specific objectives to consider.

```
solution.setObjective(0, objective[0]);
solution.setObjective(1, objective[1]);
solution.setObjective(2, objective[2]);
```

This concrete configuration is the one corresponding the many-objective variant, in charge of optimizing the cost, availability and performance. Furthermore, in order to modify the algorithm for solving two objectives, along with the command `this.setNumberOfObjectives(2)`, the user should also modify the above detailed lines to:

```
solution.setObjective(0, objective[Y]);
solution.setObjective(1, objective[Z]);
```

In which Y and Z should take different values in [0,1,2] considering that {0: cost, 1: availability, 2: performance}.

For the case of MOEA Framework, the number of objectives to optimize is also a parametrizable value which should be defined in the constructor method of both MOEA_Problem and MOEA_ProblemDiscrete, in the following way:

```
public MOEA_Problem(int number) {
     super(number, X);
}
```

Where X is the number of objectives to optimize (in this prototype, 2 or 3). After that, in order the solution to contemplate the number of objectives defined, the method evaluate() that can also be found in both MOEA_Problem and MOEA_ProblemDiscrete should contemplate a variable coined as f. This variable is represented as an array and should be filled with the objectives to consider. We depict here an example with two objectives, cost and availability.

```
double[] f = new double[2];
f[0] = this.calculateCost(mapOfElements, sol);
f[1] = -1.0*this.calculateAvailability(mapOfElements, sol);
```

On the other hand, if the number of objectives to optimize is 3, these is a possible example of implementation:

```
double[] f = new double[3];
f[0]=this.calculateCost(mapOfElements, sol);
f[1]=-1.0*this.calculateAvailability(mapOfElements, sol);
f[2]=-1.0*this.calculatePerformance(mapOfElements, sol);
```

8.4 Licensing information

Information about license not included yet.

8.5 Download

The prototype, both in its .jar version and its source code, is openly available and reachable through the GitLab provided by TECNALIA. In order to obtain all the necessary files for running the prototype described in this section, the reader should access the following link:

https://git.code.tecnalia.com/piacere/private/t53-iop-optimizer.git

The source code will be available on the public git repository and accessible through the project's website https://www.piacere-project.eu/. At the time of writing this deliverable, the source code is provided under request through an email to the address appearing on the website (https://www.piacere-project.eu/) in the footer under "Contact Us".

9 Conclusions

This document has presented a fist version of the Infrastructural elements Catalogue and the IOP. On one hand, the Infrastructural Elements Catalogue presented in PIACERE stores information about the services available at service providers as well as the instances of each of these services being used by the different applications being deployed by the PIACERE infrastructure.

On the other hand, the optimizer will use optimization algorithms, seeking for an optimized deployment configuration of the IaC on the appropriate infrastructural elements that best meet the predefined constraints. Thus, the IOP will succeed if it is able to propose the most optimized deployment configuration of the infrastructural code taking into consideration the constraints predefined. To this end, several deployment configurations will be shown and ranked.

In this document, we have described the optimization problem modelled, which have been built under the MOO paradigm, showing its adequacy with some examples. Furthermore, we have shown the prototypes for both components: the Infrastructural Elements Catalogue and the IOP In this deliverable we have also justified the MOO frameworks used, determining why we have not employed additional alternatives such as ECJ, Jenetics or JCLEC-MO. Finally, we have described the implementation details of both prototypes.

Further work on these prototypes contemplates the connection with the DOML, the development of advanced optimization methods, or the modification of the authentication protocol for the infrastructural service catalogue.

10 References

[1] D. Consortium, «D2.5 DECIDE Detailed architecture,» 2019.

[2] Gaia-X European Association for Data and Cloud AISBL;, "Gaia-X A Federated Secure Data Infrastructure," 2021. [Online]. Available: https://gaia-x.eu/. [Accessed 8 11 2021].

[3] Gaia-X European Association for Data and Cloud AISBL, , "Gaia-X Architecture Document, 21.06 Release," 2021.

[4] eco - Association of the Internet Industry, “Core Catalogue Features,” [Online]. Available: https://www.gxfs.de/federation-services/federated-catalogue/core-catalogue-features/.

[5] T. O. I. (OAI), "OpenAPI Specification v3.1.0," 15 February 2021. [Online]. Available: https://spec.openapis.org/oas/v3.1.0.html. [Accessed 08 11 2021].

[6] D. Consortium, "D7.2 Intermediate market, innovation and Applicability Analysis," 2018.

[7] Rackspace, "rackspace," [Online]. Available: https://www.rackspace.com/.

[8] Jamcracker, "https://www.jamcracker.com/," [Online]. Available: https://www.jamcracker.com/.

[9] Computenext, "https://www.computenext.com/," [Online]. Available: https://www.computenext.com/.

[10] CloudBolt, “CloudBolt,” [Online]. Available: https://www.cloudbolt.io/.

[11] Embotics, “Embotics,” [Online]. Available: http://www.embotics.com/.

[12] Morpheus, “Morpheus,” [Online]. Available: https://www.morpheusdata.com/.

[13] AppDynamics, “AppDynamics,” [Online]. Available: https://www.appdynamics.com/.

[14] BMC, “BMC's TrueSight Pulse Monitoring,” [Online]. Available: http://truesightpulse.bmc.com/.

[15] New Relic, “New Relic,” [Online]. Available: https://newrelic.com/.

[16] Nagios, “Nagios,” [Online]. Available: https://www.nagios.org/.

[17] Zabbix, “Zabbix,” [Online]. Available: http://www.zabbix.com/.

[18] OpenStack, “OpenStack Monasca,” [Online]. Available: https://www.openstack.org/.

[19] Amazon, “AWS CloudWatch,” [Online]. Available: https://aws.amazon.com/es/cloudwatch/.

[20] Jamcracker, "Jamcracker - Cloud Service Brokerage & Cloud Governance solutions," [Online]. Available: https://www.jamcracker.com/.

[21] T-Systems International GmbH;, "Which Cloud would you like? The best mix with the cloud broker," 09 2014. [Online]. Available: https://www.t-systems.com/blob/77590/d489d9858fbf2e571e33d0d69050c300/dl-flyer-cloud-broker-data.pdf.

[22] Nephos, "Cloud Brokerage - Nephos Technologies," [Online]. Available: http://www.nephostechnologies.com/content/cloud-brokerage.

[23] Ubercloud, "Ubercloud," [Online]. Available: http://uber-cloud.com.

[24] CloudMore, "CloudMore," [Online]. Available: http://web.cloudmore.com/.

[25] D. Consortium, "D7.3 Final Market Innovation and Applicability Analysis," 2019.

[26] cloudmore, "Cloudmore," [Online]. Available: https://web.cloudmore.com/.

[27] IBM, "IBM Cloud Brokerage Solutions," [Online]. Available: https://www.ibm.com/us-en/marketplace/cloud-brokerage-solutions.

[28] Nephostechnologies, "Hybrid Cloud Management," [Online]. Available: http://www.nephostechnologies.com/technology/hybrid-cloud-management/.

[29] Intercloud, "Intercloud platform," [Online]. Available: https://intercloud.com/managed-solutions/.

[30] Jamcracker, "Jamcracker," [Online]. Available: https://www.jamcracker.com/cloud-service-brokerage.

[31] S. S. M. E. &. R. S. Mahdavi, «Metaheuristics in large-scale global continues optimization: A survey,» *Information Sciences,* nº 295, pp. 407-428, 2015.

[32] R. T. &. A. J. S. Marler, «Survey of multi-objective optimization methods for engineering,» *Structural and multidisciplinary optimization,* vol. 26, nº 6, pp. 369-395, 2004.

[33] J. O. E. M. D. Y. X. S. S.-S. S. C. D. .. &. H. F. Del Ser, «Bio-inspired computation: Where we stand and what's next,» *Swarm and Evolutionary Computation,* vol. 48, pp. 220-250, 2019.

[34] X. S. C. Z. X. R. G. A. H. &. K. M. (. Yang, Swarm intelligence and bio-inspired computation: theory and applications, Newnes, 2013.

[35] L. B. M. &. T. E. G. Jourdan, «Hybridizing exact methods and metaheuristics: A taxonomy,» *European Journal of Operational Research,* vol. 199, nº 3, pp. 620-629, 2009.

[36] H. Müller-Merbach, "Heuristics and their design: a survey," *European Journal of Operational Research,* vol. 8, no. 1, pp. 1-23, 1981.

[37] M. &. P. J. Y. (. Gendreau, Handbook of metaheuristics (Vol. 2), New York: Springer, 2010.

[38] S. Nesmachnow, «An overview of metaheuristics: accurate and efficient methods for optimisation,» *International Journal of Metaheuristics,* vol. 3, nº 4, pp. 320-347, 2014.

[39] E. C. R. D. F. O. E. M. A. D. &. P. A. Osaba, «Good practice proposal for the implementation, presentation, and comparison of metaheuristics for solving routing problems,» *Neurocomputing,* vol. 271, pp. 2-8, 2018.

[40] T. T. Y. S. &. B. J. Nguyen, «Evolutionary dynamic optimization: A survey of the state of the art,» *Swarm and Evolutionary Computation,* vol. 6, pp. 1-24, 2012.

[41] A. O. Y. S. &. F. L. Gupta, «Insights on transfer optimization: Because experience is the best teacher,» *IEEE Transactions on Emerging Topics in Computational Intelligence,* vol. 2, nº 1, pp. 51-64, 2017.

[42] D. B. D. B. &. C. C. Bertsimas, «Theory and applications of robust optimization,» *SIAM review,* vol. 53, nº 3, pp. 464-501, 2011.

[43] L. C. R. N. B. M. E. D. Z. Kaced K, «Bat algorithm based maximum power point tracking for photovoltaic system under partial shading conditions,» *Solar Energy,* vol. 158, pp. 490-503, 2017.

[44] R. S. Y. X. L. K. Fister I, «Planning the sports training sessions with the bat algorithm,» *Neurocomputing,* vol. 149, pp. 993-1002, 2015.

[45] Y. X. F. I. D. S. J. L.-G. P. V.-P. A. Osaba E, «A discrete and improved bat algorithm for solving a medical goods distribution problem with pharmacological waste collection,» *Swarm and Evolutionary Computation,* 2018.

[46] C. R. Y. X. F. I. J. L.-G. P. D. S. J. Osaba E, «On efficiently solving the vehicle routing problem with time windows using the bat algorithm with random reinsertion operators,» de *Nature-Inspired Algorithms and Applied Optimization*, Springer, 2018, pp. 69-89.

[47] S. S. C. S. D. A. C. S. S. J. Dey N, «Firefly algorithm for optimization of scaling factors during embedding of manifold medical information: An application in ophthalmology imaging,» *Journal of Medical Imaging and Health Informatics,* vol. 4, nº 3, pp. 384-394, 2014.

[48] A. P. N. S. P. T. Karthikeyan S, «A hybrid discrete firefly algorithm for solving multi-objective flexible job shop scheduling problems,» *International Journal of Bio-Inspired Computation,* vol. 7, nº 6, pp. 386-4017, 2015.

[49] Y. X. D. F. O. E. M. A. P. A. Osaba E, «A discrete firefly algorithm to solve a rich vehicle routing problem modelling a newspaper distribution system with recycling policy,» *Soft Computing,* vol. 21, nº 18, pp. 5295-5308, 2017.

[50] T.-B. A. L. I. B. M. P. C. Del Ser J, «Nature-inspired heuristics for the multiple-vehicle selective pickup and delivery problem under maximum profit and incentive fairness criteria,» *IEEE Congress on Evolutionary Computation,* pp. 480-487, 2017.

[51] P. V. Karakatič S, «A survey of genetic algorithms for solving multi depot vehicle routing problem,» *Applied Soft Computing,* vol. 27, pp. 519-532, 2015.

[52] M. F. A. A. M. M. Afshar A, «State of the art review of ant colony optimization applications in water resource management,» *Water Resources Management,* vol. 29, nº 11, pp. 3891-3904, 2015.

[53] E. L. M. &. T. L. Zitzler, «SPEA2: Improving the strength Pareto evolutionary algorithm,» TIK-report, 2001.

[54] K. P. A. A. S. &. M. T. A. M. T. Deb, «A fast and elitist multiobjective genetic algorithm: NSGA-II,» *IEEE transactions on evolutionary computation,* vol. 6, nº 2, pp. 182-197, 2002.

[55] Q. &. L. H. Zhang, «MOEA/D: A multiobjective evolutionary algorithm based on decomposition,» *IEEE Transactions on evolutionary computation,* vol. 11, nº 6, pp. 712-731, 2007.

[56] N. N. B. &. E. M. Beume, «SMS-EMOA: Multiobjective selection based on dominated hypervolume,» *European Journal of Operational Research,* vol. 181, nº 3, pp. 1653-1669, 2007.

[57] L. Y. &. C. C. Tseng, «Multiple trajectory search for unconstrained/constrained multi-objective optimization,» *IEEE Congress on Evolutionary Computation,* pp. 1951-1958, 2009.

[58] A. J. D. J. J. L. F. D. B. &. A. E. Nebro, «MOCell: A cellular genetic algorithm for multiobjective optimization,» *International Journal of Intelligent Systems,* vol. 24, nº 7, pp. 726-746, 2009.

[59] R. &. I. H. Tanabe, «Review and analysis of three components of the differential evolution mutation operator in MOEA/D-DE,» *Soft Computing,* vol. 23, nº 23, pp. 12843-12857, 2019.

[60] S. &. L. J. Kukkonen, «GDE3: The third evolution step of generalized differential evolution,» *IEEE congress on evolutionary computation,* pp. 443-450, 2005.

[61] C. C. &. L. M. S. Coello, «MOPSO: A proposal for multiple objective particle swarm optimization,» *Congress on Evolutionary Computation,* pp. 1051-1056, 2002.

[62] A. J. D. J. J. G.-N. J. C. C. C. L. F. &. A. E. Nebro, «SMPSO: A new PSO-based metaheuristic for multi-objective optimization,» *IEEE Symposium on computational intelligence in multi-criteria decision-making ,* pp. 66-73, 2009.

[63] S. &. C. C. C. A. Zapotecas Martínez, «A multi-objective particle swarm optimizer based on decomposition,» *13th annual conference on Genetic and evolutionary computation,* pp. 69-76, 2011.

[64] J. D. &. C. D. W. Knowles, «M-PAES: A memetic algorithm for multiobjective optimization,» *Congress on Evolutionary Computation,* pp. 325-332, 2000.

[65] K. &. J. H. Deb, «An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints,» *IEEE transactions on evolutionary computation,* vol. 18, nº 4, pp. 577-601, 2013.

[66] S. &. Y. S. Jiang, «A strength Pareto evolutionary algorithm based on reference direction for multiobjective and many-objective optimization,» *IEEE Transactions on Evolutionary Computation,* vol. 21, nº 3, pp. 329-346, 2017.

[67] S. B. A. A. &. K. Ö. A. Özdemir, «Multi-objective evolutionary algorithm based on decomposition for energy efficient coverage in wireless sensor networks,» *Wireless personal communications,* vol. 71, nº 1, pp. 195-215, 2013.

[68] J. &. Z. E. Bader, «HypE: An algorithm for fast hypervolume-based many-objective optimization,» *Evolutionary computation,* vol. 19, nº 1, pp. 45-76, 2011.

[69] R. &. C. C. C. A. Hernández Gómez, «Improved metaheuristic based on the R2 indicator for many-objective optimization,» *Annual Conference on Genetic and Evolutionary Computation,* pp. 679-686, 2015.

[70] C. A. C. C. &. P. G. T. Coello, «A micro-genetic algorithm for multiobjective optimization,» *International conference on evolutionary multi-criterion optimization,* pp. 126-140, 2001.

[71] J. S. K. O.-E. L. B. L. W. M. E. M. .. &. D. S. Alonso, «DECIDE: An Extended DevOps Framework for Multi-cloud Applications,» *3rd International Conference on Cloud and Big Data Computing,* pp. 43-48, 2019.

[72] M. T. A. B. M. N. &. D. S. J. Arostegi, «A heuristic approach to the multicriteria design of IaaS cloud infrastructures for Big Data applications,» *Expert Systems,* vol. 35, nº 5, p. e12259, 2018.

[73] I. A. T. Y. I. A. N. B. M. S. G. A. &. K. S. U. Hashem, «The rise of "big data" on cloud computing: Review and open research issues,» *Information systems,* vol. 47, pp. 98-115, 2015.

[74] M. A. Rappa, «The utility business model and the future of computing services,» *IBM systems journa,* vol. 43, nº 1, pp. 32-42, 2004.

[75] H. D. F. &. B. S. Herodotou, «No one (cluster) size fits all: automatic cluster sizing for data-intensive analytics,» *2nd ACM Symposium on Cloud Computing,* pp. 1-14, 2004.

[76] G. &. S. P. Horn, «MELODIC: utility based cross cloud deployment optimisation,» *32nd International Conference on Advanced Information Networking and Applications Workshops,* pp. 360-367, 2018.

[77] A. A. T. C. A. P. a. S. C. S. B. K. Dewangan, «Extensive review of cloud resource management techniques in industry 4.0: Issue and challenges,» *Software: Practice and Experience,* 2020.

[78] J. S. a. J. S. N. Khattar, «Toward energy-efficient cloud computing: a survey of dynamic power management and heuristics-based optimization techniques,» *Journal Supercomputing,* vol. 75, nº 8, p. 4750–4810, 2019.

[79] K. H. Y. H. a. X. T. L. Zhu, «A Self-Adapting Task Scheduling Algorithm for Container Cloud Using Learning Automata,» *IEEE Access,* 2121.

[80] Y. Z. a. L. Z. W. Zhu, «A three-dimensional virtual resource scheduling method for energy saving in cloud computing,» *Future Generation Computer Systems,* vol. 69, p. 66–74, 2017.

[81] R. N. C. A. V. V. J. B. a. R. O. S. Y. Zhao, «SLA-Aware and Deadline Constrained Profit Optimization for Cloud Resource Management in Big Data Analytics-as-a-Service Platforms,» *12th International Conference on Cloud Computing ,* p. 146–155, 2019.

[82] G. P. G. D. A. E. D. N. M. L. a. M. A. A. d. S. M. Ciavotta, «Architectural Design of Cloud Applications: a Performance-aware Cost Minimization Approach,» *IEEE Transactions on Cloud Computing,* 2020.

[83] L. R. M. W. a. B. H. F. S. Mireslami, «Dynamic Cloud Resource Allocation Considering Demand Uncertainty,» *IEEE Transactions on Cloud Computing,* 2019.

[84] M. C. a. D. A. E. Gianniti, «Optimizing Quality-Aware Big Data Applications in the Cloud,» *IEEE Transactions on Cloud Computing,* 2018.

[85] Z. Á. Mann, «Resource Optimization Across the Cloud Stack,» *IEEE Transactions on Parallel and Distributed Systems,* vol. 29, nº 1, p. 169–182, 2018.

[86] P. O. a. P. Nawrocki, «Resource Usage Cost Optimization in Cloud Computing Using Machine Learning,» *IEEE Transactions on Cloud Computing,* 2020.

[87] L. R. B. H. F. a. M. W. S. Mireslami, «Simultaneous Cost and QoS Optimization for Cloud Resource Allocation,» *IEEE Transactions on Network and Service Management,* vol. 14, nº 1, p. 676–689, 2017.

[88] J. E. F. M. a. E. D. N. B. Xiang, "Joint Network Slicing and Mobile Edge Computing in 5G Networks," *IEEE International Conference on Communications,* pp. 1-7, 2019.

[89] J. T. a. Y. L. C. Li, «Service cost-based resource optimization and load balancing for edge and cloud environment,» *Knowledge Information Systems,* vol. 62, nº 11, p. 4255–4275, 2020.

[90] J. L. B. C. a. C. W. Y. Li, «Joint Optimization of Radio and Virtual Machine Resources With Uncertain User Demands in Mobile Cloud Computing,» *IEEE Transactions on Multimedia,* vol. 20, nº 9, p. 2427–2438, 2018.

[91] M. Z. a. A. H. A. B. M. B. Alam, «A Reliability-Based Resource Allocation Approach for Cloud Computing,» *7th International Symposium on Cloud and Service Computing,* p. 249–252, 2017.

[92] B. C. a. X. L. Y. Yue, «Resource Optimization and Delay-aware Virtual Network Function Placement for Mapping SFC Requests in NFV-enabled Networks,» *13th International Conference on Cloud Computing,* p. 267–274, 2020.

[93] W. L. A. G. a. Y. M. Z. Xu, «Throughput maximization and resource optimization in NFV-enabled networks,» *International Conference on Communications ,* pp. 1-7, 2017.

[94] P. H. K. X. a. j. P. D. Li, «Virtual Network Function Placement Considering Resource Optimization and SFC Requests in Cloud Datacenter,» *IEEE Transactions on Parallel and Distributed Systems,* vol. 29, nº 7, p. 1664–1677, 2018.

[95] E. C. R. D. F. O. E. M. A. D. &. P. A. Osaba, «Good practice proposal for the implementation, presentation, and comparison of metaheuristics for solving routing problems,» *Neurocomputing,* vol. 271, pp. 2-8, 2018.