Multi-criteria Optimization for Resource Allocation in Multi-access Edge Computing

Nour-El-Houda Yellas

HESAM UNIVERSITÉ

le cnam

ÉCOLE DOCTORALE SCIENCES DES MÉTIERS DE L'INGÉNIEUR
Centre d'études et de recherche en informatique et communications

THÈSE

présentée par : **Nour El-Houda YELLAS**

soutenue le : **18 décembre 2023**

pour obtenir le grade de : **Docteur d'HESAM Université**

préparée au : **Conservatoire national des arts et métiers**

Discipline : **Informatique**

Spécialité : **Informatique**

Optimisation Multi-critères de l'Allocation des Ressources de Calcul à la Bordure du Réseau

THÈSE dirigée par :
M^me^ BOUMERDASSI Selma MCF HDR, Cnam

et co-encadrée par :
M. SECCI Stefano Professeur, Cnam

Jury

M. André-Luc BEYLOT	Prof., IRIT, ENSEEIHT	Rapporteur
M. Yassine HADJADJ-AOUL	Prof., IRISA/Inria, Univ. de Rennes	Rapporteur
M^me^. Anne FLADENMULLER	Prof., LIP6, Sorbonne Université	Examinatrice
M^me^ Bernardetta ADDIS	Prof., LORIA, Univ. de Lorraine	Examinatrice
M^me^ Thi-Mai-Trang Nguyen	Prof., L2TI, Sorbonne Paris Nord	Examinatrice
M. Khaled BOUSETTA	Prof., L2TI, Sorbonne Paris Nord	Examinateur

THÈSE

HESAM UNIVERSITÉ

le cnam

ÉCOLE DOCTORALE SCIENCES DES MÉTIERS DE L'INGÉNIEUR
Centre d'études et de recherche en informatique et communications

THÈSE

présentée par : **Nour El-Houda YELLAS**

soutenue le : **18 décembre 2023**

pour obtenir le grade de : **Docteur d'HESAM Université**

préparée au : **Conservatoire national des arts et métiers**

Discipline : **Informatique**

Spécialité : **Informatique**

Multi-criteria Optimization for Resource Allocation in Multi-access Edge Computing

THÈSE dirigée par :
M^me^ BOUMERDASSI Selma MCF HDR, Cnam

et co-encadrée par :
M. SECCI Stefano Professeur, Cnam

Jury

M. André-Luc BEYLOT	Prof., IRIT, ENSEEIHT	Rapporteur
M. Yassine HADJADJ-AOUL	Prof., IRISA/Inria, Univ. de Rennes	Rapporteur
M^me^. Anne FLADENMULLER	Prof., LIP6, Sorbonne Université	Examinatrice
M^me^ Bernardetta ADDIS	Prof., LORIA, Univ. de Lorraine	Examinatrice
M^me^ Thi-Mai-Trang Nguyen	Prof., L2TI, Sorbonne Paris Nord	Examinatrice
M. Khaled BOUSETTA	Prof., L2TI, Sorbonne Paris Nord	Examinateur

THÈSE

Abstract

Mobile cloud computing was a promising paradigm to achieve gigabit access in the new-generation wireless networks by offloading the computation tasks of mobile devices to a central cloud data center. However, bandwidth congestion at the core network is an inherent limitation in terms of communication delays and energy consumption. Multi-access edge computing (MEC) was initially proposed to overcome the limitation of mobile cloud computing. It moves the cloud computing data center down into the access network to better meet the requirements of pervasive applications as the set of constraints in terms of quality of service can go beyond basic low-latency, high bandwidth and real-time access to radio-network information. However, colocating application servers, cellular core network and radio-access network sub-components at MEC hosts makes the resource assignment task a very important challenge to tackle as it has direct impact on network operations cost and end-to-end infrastructure reliability.

This thesis investigates a set of robust resource scheduling solutions in the context of MEC. More precisely, we address different techniques to assign the available resources on a MEC host to a given task. We focus on the optimization of the resource orchestration decisions at both the infrastructure and the application levels. First, we address the base station to MEC hosts assignment orchestration decision while taking into account a data-driven assignment objective against traffic fluctuation. Then, we go beyond MEC infrastructure management by proposing an In-network control scheme for efficiently placing MEC applications under stringent time constraints where a federated learning-based anomaly detection is considered.

Keywords: MEC (Multi-access Edge Computing), resource orchestration, optimization, 5G, virtualization, mobile network.

Résumé

L'informatique mobile combinée avec le cloud computing (ou le mobile cloud computing) consiste à décharger les tâches de calcul des appareils mobiles vers un centre de données central. Ce paradigme était prometteur pour obtenir des accès en gigabits dans les réseaux mobiles de nouvelle génération. Toutefois, la congestion de la bande passante au niveau du réseau coeur constitue une limitation inhérente en termes de temps de latence et de consommation d'énergie. L'informatique de périphérie connue aussi sous le nom du edge computing multi-accès ou MEC a été initialement proposée pour surmonter les limites du mobile cloud computing. Elle permet de déplacer les capacités de calcul des centres de données centralisées vers le réseau d'accès afin de mieux répondre aux exigences des applications omniprésentes, nécessitant des contraintes de qualité de service qui peuvent aller au-delà d'une faible latence ou d'une large bande passante. Toutefois, la colocalisation des serveurs d'application, du réseau coeur et des sous-composants du réseau d'accès radio dans les serveurs de périphérie rend l'allocation de ressources un défi très important à relever vu son impact direct sur le coût d'exploitation du réseau et sur la fiabilité de bout en bout de l'infrastructure.

Cette thèse étudie les solutions robustes d'allocation des ressources dans le contexte du edge computing multi-accès. Plus précisément, nous abordons différentes techniques pour affecter les ressources disponibles sur un ensemble d'hôtes MEC à une tâche donnée. Nous nous concentrons sur l'optimisation des décisions d'orchestration des ressources au niveau infrastructure et niveau applicatif. Tout d'abord, nous abordons le défi de l'orchestration de l'affectation des stations de base aux hôtes MEC, tout en tenant compte d'un objectif d'affectation basé sur les données contre la fluctuation du trafic généré par les utilisateurs mobiles. Ensuite, nous allons au-delà de la gestion de l'infrastructure MEC en proposant un schéma de contrôle en réseau pour placer efficacement des applications MEC sous des contraintes temporelles très strictes. Plus précisément, un système de détection d'anomalies basé sur l'apprentissage fédéré est utilisé.

Keywords : informatique de périphérie, allocation de ressources, optimisation, infrastructure, virtualisation, 5G, réseaux mobiles.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Contents

1.1 Multi-access Edge Computing paradigm

The Multi-access Edge Computing (MEC) paradigm was initially developed for running IT services close to end devices in order to lower latency and improve user experience. Despite the fact that MEC infrastructure hosts can be densely distributed at the edge, resource limitation and robustness against traffic fluctuations are important challenges to handle by network service providers. New technologies such as Network Function Virtualization (NFV) and Software Defined Networking (SDN) were proposed in the last decade; their consideration into network architecture design is to push the technology barriers toward virtualized infrastructures at Radio Access Network (RAN) subsystems as well [1], including both the centralization of the control and the virtualization and softwarization of all involved network functions.

Radio function virtualization leads to additional flexibility in a segment historically more rigid than core networks, due to the lower importance of routing in these environments. This flexibility can help to meet the growing and unpredictable demands of mobile users, and also allows the use of standard hardware to reduce costs for MNO and delay capital expenditures. In addition, MEC technology allows to cope with users demand variation, since network reconfiguration becomes an easier operation to perform [2]. Indeed, while MEC infrastructures are recognized as a 5G key enabler, the reverse is also true: 5G can be considered a key enabler for MEC infrastructures, thanks to NFV technology [3]. The deployment of virtualization facilities in the access network, for 5G functions and RAN functions, can therefore favor the deployment of MEC infrastructure elements. The so-favored deployment of application servers near end users can increase user bit rates and reduce end-to-end latency [4]. Note that MEC needs a virtualization platform to deploy its applications at the edge. In that case, the NFV platform can be used to deploy Virtual Network Functions (VNFs) and MEC applications.

The location of MEC hosts is currently envisioned by telecom operators to happen at so-called Central Offices (CO) and/or Points of Presence (PoP). The distribution of MEC hosts horizontally across different access network segments and vertically at different layers of the backhauling network is needed to meet the access latency and reliability requirements. Typically, MEC hosts are meant to be therefore situated between base stations and the core network [5]. A reference representation of the MEC infrastructure is in Figure 8.1. Strictly speaking [6], a ‘MEC host’ (cloudlet or MEC

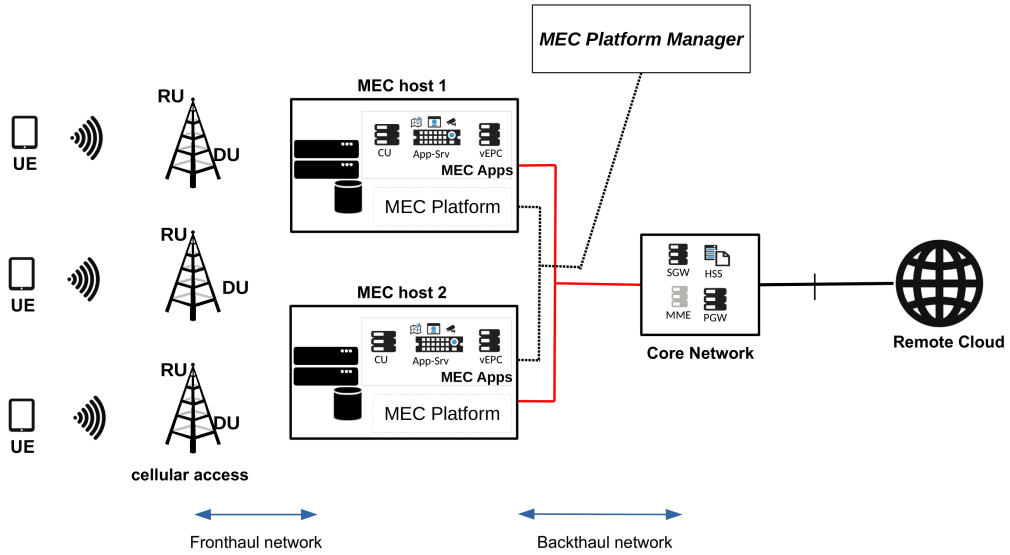


Figure 1.1: Representation of the reference MEC infrastructure.

facility) refers to the hardware servers belonging to the virtualization infrastructure; it can be generic or NFVI based, and in this case the MEC host can be deployed as a VNF, possibly supporting network slicing [3]. The 'MEC platform' is responsible for managing MEC applications. When different MEC hosts are deployed on the access network of an operator, it forms a distributed cloud referred to as 'MEC system'.

In order to have a complete control of service deployment at the MEC infrastructure, the ETSI standards call for the development of orchestration service elements, with the aim of efficiently managing the available resources on MEC hosts. Hence, the automation of the aforementioned task is deemed as one of the important challenges to address. Furthermore, as end users have the moving functionality within the mobile network, mobility support is another ETSI-MEC requirement to ensure the continuity of services. MEC applications can be then divided into state-dependant and state-independent. The former is specific to the user where part or all information should be maintained whereas the latter is not related to user activity.

Also, three major categories of MEC use cases were identified by ETSI [7]. The consumer-oriented applications, they are directly related to the end user (user equipment) such as gaming and augmented reality. The operator oriented applications, which are services that may not be directly related to users but to third parties such as safety and security applications, end-device tracking, etc. Finally, the Quality of Experience (QoE) improvement applications that aim to enhance user experience while focusing on network optimization. As examples, one can cite content caching, MEC hosts deployment optimization and resource scheduling. It is worth mentioning that this thesis focuses on the third category of use cases. In the following, we present the different challenges encountered in MEC environments.

1.2 Problem Statement and Challenges

MEC applications usually have a set of requirements such as computing capacity, energy efficiency and latency. As MEC systems are deployed on servers with limited capacities (i.e., storage and processing capacities), the availability of resources may change over time which requires migration of MEC applications from one MEC host to another. Note that each location may have a different cost in terms of performance, deployment, or both. Thus, deploying a MEC application at the best location (i.e., closest MEC host with enough resources) may not always be the best choice. To that end, MEC systems should be able to decide on the placement of MEC applications while taking into consideration that the placement decision may change over time as the conditions evolve.

In this thesis, we consider issues related to resource scheduling in MEC. In general, resource scheduling refers to the techniques used to assign a set of available resources to mobile users to accomplish a specif task at a given moment. Designing efficient resource scheduling strategies is twofold: (i) achieving the desired QoS at the user level, and (ii) optimizing costs at the edge service provider level. Resource scheduling problems in edge environments are the subject of a great deal of interest in the literature. For example, [8] discusses the research works that are related to resource scheduling in edge computing. It classifies the possible actions in resource scheduling into three categories: (i) computation offloading [9], (ii) resource orchestration [10] and (iii) resource provisioning [11]. Computation offloading is a promising solution to release User Equipment (UE) from computation-intensive tasks. The offloading decision is taken based on several requirements such as latency, cost and energy consumption. As presented by [8], the computation offloading can be

either classified based on the offloading direction i.e., UE-to-edge, edge-to-cloud, edge-to-edge, ... etc, or based on the granularity, i.e., tasks partially or totally offloaded. Another research issue in MEC environment is resource orchestration. It consists of flexibly allocating computation, communication or storage resources in order to guarantee a given Quality of Service (QoS) and can also jointly consider several resources. Note that in this thesis, resource orchestration refers to resource allocation, service and application placement or both. Finally, resource provisioning consists of allocating the suitable amount of resources to ensure QoS requirements. Note that the difference between resource orchestration and resource provisioning is that the former assigns available resources to users to ensure a given service whereas the latter ensures the availability of resources when needed.

1.3 Research Questions and Contributions

The emergence of the new pervasive applications with strict and heterogeneous requirements in terms of latency and bandwidth led to the appearance of the MEC paradigm to cope with the limitation of traditional cloud computing. On the other hand, virtualization paradigms such as NFV and SDN were originally proposed to efficiently manage the available resources leading to the appearance of network slicing concept to meet the specific and heterogeneous requirements of applications and services. Hence, designing applications-tailored slices, efficient resource allocation algorithms and resilient network and system management are now considered as the new research challenges to address in the network and telecommunication field.

The goal of this thesis is to automate and optimize the allocation of resources in a 5G-MEC environment taking into account scalability issues. The contributions of this thesis aim at answering the following research questions:

1. *From a MEC infrastructure point of view*: in disaggregated RAN where radio processing functions are split into Centralized Unit (CU) and Distributed Unit (DU), how to efficiently decide which group of Base Stations (BSs) should be served by a given CU based on MEC hosts capacity, access latency and deployment costs? Also, how could Mobile Network Operators (MNOs) make use of data analytics to optimize the orchestration solution by making it scalable and robust against near-real time deployment? These two questions are addressed in Chapters 3 and 4.

2. *From a MEC applications point of view*: when deploying distributed AI/ML (Artificial Intelli-

gence/Machine Learning) models at the edge, how to control end-to-end learning time variation due to heterogeneous resources at MEC hosts?
This research question is detailed in Chapters 5 and 6.

This thesis aims at providing a deep understanding of the resource orchestration challenges in MEC environments while proposing several novel approaches that efficiently address them from different levels (i.e., infrastructure level and application level). The main contributions are presented in the following.

1.3.1 MEC infrastructure level

1.3.1.1 Spatial clustering for BS-to-MEC hosts assignment

In this work, we focus on optimizing MEC orchestration tasks to deal with new 5G pervasive applications where we consider the complexity and scalability of base stations to MEC servers assignment problem. We address this challenge to include secondary objectives to existing algorithms for MEC orchestration, and in particular for the problem of finding assignments of base stations to MEC hosts. The proposed framework is composed of two steps. We first apply a spatial clustering model on the set of BSs at the preprocessing phase then we solve the assignment problem of the resulting clusters of BSs to the available MEC hosts by adapting the orchestration model. Note that the assignment operation comes at a cost defined by the access latency. We also consider the reallocation of resources where a Virtual Machine (VM) serving a user or a set of users is migrated from a MEC host to another.

The spatial clustering model groups together BSs based on their profile of traffic demands (i.e., spatio-temporal behavior) in order to minimize the variance in traffic demands within each cluster of BSs. The main goal of this proposal is to reduce the spatial and temporal complexity (i.e., the execution time and the memory consumption). The obtained results from extensive simulation against real traffic demands show how our proposal reduces time and space complexity w.r.t the baseline algorithm. Even though the proposed technique yields additional costs, its robustness allows to run the framework in a real-time manner. More results are available in Chapter 3.

1.3.1.2 Robust clustering models for BS-to-MEC hosts assignment

While the BS clustering model previously presented increases the users costs in terms of latency and since 5G services aims at ensuring utlra-low latency, we propose to extend the previous work to enhance the orchestration decision in terms of latency and the degree of controlling the MEC hosts capacity by reducing the capacity violation that may occur due to traffic fluctuation. To do so, we propose to group BSs in pairs based on a set of criteria. The corresponding work is detailed in Chapter 4. The proposed custering models are evaluated using a real-world dataset. Note that the orchestration framework is evaluated in an offline setting under different parameters while varying the number and capacity of available MEC hosts.

1.3.2 MEC application level

1.3.2.1 Optimal placement of MEC applications for a federated learning framework

As MEC paradigm allows to bring resources for AIML computing to the edge network where data to be processed is located, we propose to extend the aforementioned work to optimize resource allocation decisions at the MEC application level. We consider that the AIML model is deployed as a MEC application. Thus, we address the problem of placing Artificial Intelligence Functions (AIFs) running federated learning against connect-compute network infrastructure monitoring data, for environments where the introduction of edge computing comes with a heterogeneous and large set of computing and networking elements, requiring low latency performance. In particular, we use as reference use-case the Federated Learning (FL) anomaly detection AIF proposed in [12], adapted for the 5G infrastructure. This federated learning framework makes use of a federated learning server AIF, and a variable number of edge AIFs: the learning task is distributed to edge AIFs by load-balancing monitoring data among them, where edge AIFs interact via the server for learning model updates.

The targeted objective is to reduce the end-to-end learning time in order to respect the time threshold imposed by the specification of the application. To do so, we focus on the placement of AIFs making use of HardWare Acceleration (HWA). We model the behavior of federated learning and related inference point to guide the placement decision, taking into consideration the specific constraint and the empirical behavior of a virtualized infrastructure anomaly detection use-case. Besides hardware acceleration, we consider the specific training time trend when distributing the

training over a network, by using empirical piece-wise linear distributions and we model the placement problem as a Mixed-Integer Linear Programming (MILP). Simulation results show the impact that hardware acceleration can have in the decision of the number of deployed AIFs, while dividing by a relevant factor the distributed training time. More details about this contribution are available in Chapter 5.

1.3.2.2 MEC application placement for controlling stragglers in a federated learning environment

We extend the aforementioned placement model where we introduce the In-network Federated Learning Control (IFLC), that is an adaptive scheme for the usage of HWAs in distributed systems to compensate for end-to-end network and learning delays variations leading to stragglers. We go beyond the existing work, reformulating the model to control stragglers, defining a refined end-to-end training latency modeling, and proposing a polynomial optimal resolution algorithm hence supporting near-real-time orchestration of FL-AIFs. Through extensive performance evaluation, we highlight the impact of using hardware accelerators in meeting a higher ratio of FL participants that positively contribute to the learning effort while this ratio is increased by up to 100% with respect to a first-fit algorithm. More details are available in Chapter 6.

Finally, Figure 1.2 outlines the contributions of this thesis.

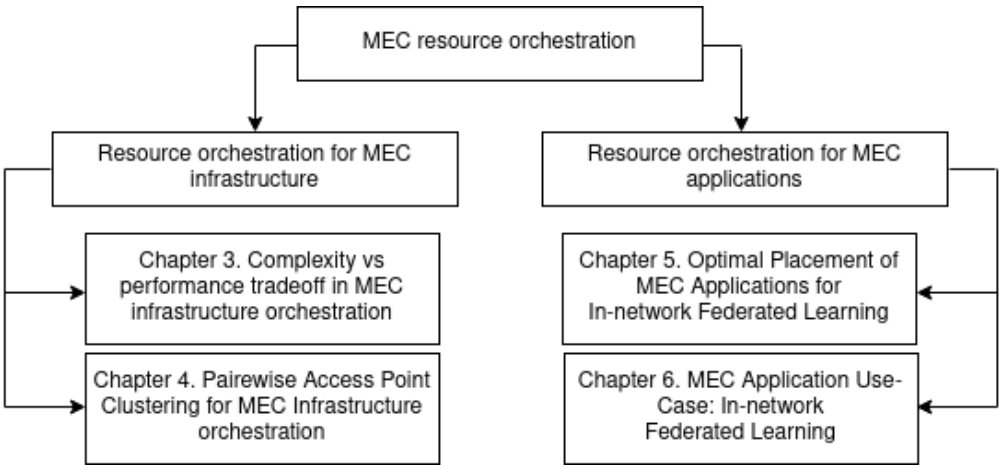


Figure 1.2: Thesis contributions.

1.4 Thesis Outline

In this chapter, we gave a brief overview of MEC paradigm and the different encountered challenges in this type of architecture. Also, the contributions to this research field were summarized.The rest of this thesis is organized as follows.

- The next chapter provides a background overview on the MEC paradigm and the existing efforts in the literature that are related to resource scheduling in MEC with a focus on resource orchestration challenges.
- Our first contribution on MEC infrastructure orchestration is presented in Chapter 3 where a spatial clustering model is integrated to an adapted BS-to-MEC assignment model.
- An extension of the previous work is presented in Chapter 4 where a collection of BS clustering models are introduced to target lower latency costs with a higher control over MEC hosts capacity.
- Chapter 5 presents a MEC orchestration solution at the application level where we propose a placement model of the artificial intelligence functions running in a federated learning setting at the edge of the network. The goal is to reduce the overall learning time targeting a time limitation imposed by the application requirements.
- Chapter 6 builds on top of the previous work by proposing a new in-network control schema for placing the AIFs with a fine-grained decomposition of time components.
- Finally, conclusions and possible considerations for future works are drawn in Chapter 7.

Note that each chapter has it own list of notations that are independent of the other chapters.

1.5 List of Publications

- N. -E. -H. Yellas, S. Boumerdassi, A. Ceselli and S. Secci, "Complexity-Performance Trade-offs in Robust Access Point Clustering for Edge Computing," 2021 17th International Conference on the Design of Reliable Communication Networks (DRCN), Milano, Italy, 2021, pp. 1-8, doi: 10.1109/DRCN51631.2021.9477332, Best Paper Award.

- N. -E. -H. Yellas, S. Boumerdassi, A. Ceselli, B. Maaz and S. Secci, "Robust Access Point Clustering in Edge Computing Resource Optimization," in IEEE *Transactions on Network and Service Management*, vol. 19, no. 3, pp. 2738-2750, Sept. 2022, doi: 10.1109/TNSM.2022.3186856.

- N. -E. -H. Yellas, B. Addis, R. Riggio and S. Secci, "Function Placement and Acceleration for In-Network Federated Learning Services," 2022 18th International Conference on Network and Service Management (CNSM), Thessaloniki, Greece, 2022, pp. 212-218, doi: 10.23919/CNSM55787.2022.9964625.

- S. B. Ruba, N. E. -H. Yellas and S. Secci, "Anomaly Detection for 5G Softwarized Infrastructures with Federated Learning," 2022 1st International Conference on 6G Networking (6GNet), Paris, France, 2022, pp. 1-4, doi: 10.1109/6GNet54646.2022.9830390.

- C. -D. Phung, N. -E. -H. Yellas, S. B. Ruba and S. Secci, "An Open Dataset for Beyond-5G Data-driven Network Automation Experiments," 2022 1st International Conference on 6G Networking (6GNet), Paris, France, 2022, pp. 1-4, doi: 10.1109/6GNet54646.2022.9830292.

Submitted for publication

- N-E-H. Yellas, B. Addis, S. Boumerdassi, R. Riggio, S. Secci, "In-network Federated Learning Control," submitted for publication in IEEE *Transactions on Network and Service Management.*

Chapter 2

State of the art

Contents

2.1 Introduction

Resource scheduling in edge computing has attracted widespread interest from both industry and academia. This chapter overviews the state-of-the-art works related to MEC resource scheduling from various perspectives. Specifically, we present in section 2.2 works combining MEC and virtualization technologies while covering research efforts in data-driven MEC infrastructure orchestration. For instance, MEC and Virtualized Radio Access Network (vRAN) are complementary technologies where parts of the RAN protocol stack can be virtualized at the MEC host [13]. The challenge of assigning base stations to MEC hosts where some RAN functions are deployed is tackled in this section. MEC infrastructure planning is addressed as well. Section 2.3 elaborates the orchestration of MEC applications where a special focus is given to FL based-AI applications with low-latency constraints, as well as one of the most challenging aspect in FL environments (i.e., straggling effect). Section 2.4 draws conclusions.

2.2 Access Points Clustering for MEC Infrastructure Orchestration

In the following, we provide the necessary background on virtualization in edge computing infrastructures, network analytics and optimization.

2.2.1 Network Virtualization

MEC is one of the 5G key enabler technologies whose main goal is to reduce access latency and optimize bandwidth to provide real time performance. Combining it with NFV technology can be of a great benefit for MNOs since the management operations can be held by the NFV architecture, more precisely by the NFV Management and Orchestration (MANO) subsystem [6]. Several works exist in the area of MEC-NFV MANO, proposing algorithms or architectures taking advantage from the presence of MEC-NFV systems. For instance, [14] addresses the relationship between MEC and other technologies that are considered as 5G enablers such as NFV and SDN: the authors propose an architectural framework where an SDN controller is responsible for management operations in a MEC-NFV environment, hence being able to reconfigure the network stack to take into consideration orchestration decisions such as the assignment of APs to MEC hosts. Other works focus on VNF placement in a MEC environment [15, 16], balancing the placement across multiple MEC locations. A

clustering scheme for network service chaining is proposed in [17] in order to minimize end-to-end service latency in MEC. More details about the MEC architecture and different orchestration and deployment scenarios are presented in [10].

In [18], a study was conducted on how a MEC infrastructure should be planned, that is, where MEC facilities should be placed, as a function of different MEC resource placement policies. A take-away result is that for a large metropolitan area network as the one of Paris, France, the number of MEC facilities ranges from 5 to 20. The workload was equivalent to plan for as much as one virtual machine per mobile user, which can be considered as an upper bound, and that for a network of approximately 180 thousands users with 606 BSs. The authors used real data volume information from a french mobile network.

From a radio-access perspective, architectures have evolved toward the virtualization and disaggregation of its control-plane and data-plane functions to improve interference coordination and resource efficiency. This evolution started with the Centralized/Centralized Radio Access Network (C-RAN), in 2010, where the innovation consisted in disaggregating Access Point (AP) facilities into two main units: Radio functions assured by Remote Radio Heads (RRHs) that are deployed on cell sites, and Base Band computation functions provided by Base Band Units (BBUs).
BBUs are then centralized at so-called BBU pools, hence taking advantage from the centralization for resource allocation and scaling [19]. More recently, the C-RAN evolution has been integrated in 5G systems, where a more dense deployment of base stations is needed for a more flexible infrastructure, leading to a generalized virtualized or software-defined RAN (vRAN or SD-RAN) environment. In vRAN, the equivalent of the BBU function can be split into two units, the CU and the DU, in order to facilitate the virtualization and radio scheduling tasks [20], while the radio part is called Radio Unit (RU). Splitting radio processing functions is known as 'functional split' [19] and it enables to choose the functions that turn on cell sites and those that will be offloaded to CUs, with different splitting options [21], possibly in a dynamic (runtime) and flexible (different options decisions for different segments and times) fashion.

Many works investigate on how to combine vRAN and MEC technologies [13]. In [22] the authors implement a MEC platform on the vRAN front-haul, and evaluate the QoS for end users for two different locations of MEC hosts. Authors in [1] propose a MEC vRAN joint design problem, introducing an optimization framework that aims to simultaneously find the best functional split of

BSs and MEC service placement, taking into account flow routing. The integration of vRAN with SD-x system lead to the Open RAN (O-RAN) initiative, which has the goal to disaggregate software and hardware and to create open interfaces for more flexibility. Many O-RAN software releases exist today already [23]. In [24], the authors discuss the RAN evolution including the detailed description of the O-RAN reference architecture.

2.2.2 Data-driven MEC Orchestration

Given the natural limitations of MEC hosts in terms of computing resources, resource orchestration is an important task to optimize its utilization, particularly important when considering the environmental footprint of edge computing [25]. Thus, operations that consist of re-assigning APs to other MEC hosts need to be deployed; this is needed to ensure that a number of desirable Key Performance Indicators (KPIs) are met, as for instance maximizing resource utilization, increasing resiliency against network and computing impairments, and increasing robustness against load variations in time and space.

Often, data analytics techniques are used in MEC design frameworks, so as to take into account dynamic load and communication channel states. For instance, stream or online data-analytics is needed in mobile computation offloading frameworks, where tasks offloading online decisions need to be made. In this direction, a feedback prediction model of average resource usage (RAM and Central Processing Unit (CPU)) and offloading time is proposed in [26]. In [27], the authors tackle the offloading decision for MEC applications where the performance is evaluated using real-world dataset. Reference [28] aims at offloading intensive computing tasks for energy saving by optimizing resource allocation, and [29] presents solutions for computation offloading in edge servers for internet of connected vehicles. Near-real-time or offloading data analytics is indeed often considered when addressing MEC design problems.

Leveraging on network data analysis is a common requirement for clustering techniques in C-RAN and MEC environments. In [30] [31] the authors propose a clustering scheme for APs, where APs of each cluster share the same data processing units that are centralized in datacenters to optimize costs and energy consumption in vRAN. Another example is [32], where the authors aim at predicting mobile traffic generated by a cluster of APs to anticipate MEC resource orchestration using real-world dataset. In [33] the authors propose an AP geo-clustering technique, while taking into account the

spatial distribution of mobile traffic. The main goal is to define MEC clusters as a set of AP and users served by the same MEC host, so that the whole area is partitioned into MEC clusters, in order to offload the core network by maximizing intra MEC hosts communications. Similarly, in [34] where the authors apply the temporal clustering model proposed in [35] on traffic demands of a real-world dataset, and integrate it into an orchestration model; the temporal clustering consists of grouping together similar mobile network profiles using the traffic volume generated by APs at a time slot: this allows to retrieve a reduced number of profiles, with a similarity assessment based on traffic volume and traffic distribution.

Recently, the authors in [36] propose an access point clustering scheme extending K-means to use 3D Hyperbolic distance, using access points locations and traffic demands; the algorithm groups together APs with complementary demands behaviors, that is, not grouping together APs with similar demand behavior to avoid idle states during off-peak hours. A similar approach is presented in [37], where the goal is to reduce the number of reconfiguration handovers, i.e., change of BBUs for base stations, also called later in this thesis as switching operation.

2.2.3 Summary

The standpoint we adopt in this thesis is the one of an operator running a MEC infrastructure the operator leverages on, for converging MEC applications and virtualized network functions. Hence APs are assigned to MEC hosts facilities in a dynamic way by means of MANO operations, leveraging on a programmable network stack between APs and MEC infrastructure, hence going largely beyond the legacy situation where APs are statically assigned to COs and PoPs. In our work, we therefore do not need to delve into the details related to, for instance, functional splitting and the actual coexistence of NFV and MEC systems; on the other hand, our model has to take into consideration the traffic fluctuations deriving from the AP to MEC host assignments and related MEC switching operations. Among all the previous works, either spatial diversity or load changes over time are considered in the clustering and AP-to-cloud facility assignment modeling. Taking both the spatial and temporal dimensions is however often not explicitly modeled. The resolution approach in [34] does model both time and space dimensions when determining an assignment plan by means of an optimization model; the approach consists in applying decomposition techniques, to obtain an extended formulation which is then optimized by a branch-and-price algorithm.

2.3 Placement of MEC applications

Network softwarization technologies made their way into access networks in such a way that not only nowadays network functions are already mostly deployed as virtualized nodes, but also hardware components, for radio and computing systems, are redesigned to be re-programmable by external software and dynamically allocated and shared. The derived landscape is therefore a natural application domain for artificial intelligence, because many new decision making points appear and many monitoring probes are made available to network and service management systems. In the following, we review recent works in the area of AI integration to networks, with a particular focus on federated learning applications.

2.3.1 In-network AI applications

Incorporating artificial intelligence and machine learning techniques in networks can be beneficial for a high number of applications [38]. For instance, authors in [39] propose a neural network-based framework for Service Level Agreement (SLA) management in an SDN-NFV environment. In [40], an autonomic SLA enforcement strategy is proposed for a cloud environment, with a closed loop system to map low-level metrics to high-level SLA objectives.

Another AIML application is anomaly detection and fault management, which consists in detecting abnormal network states, localizing the root cause and then proposing a remediation action to comeback to a normal working condition. In [41], the authors proposed a centralized AIML framework making use of autoencoders to detect anomalies at different infrastructure levels; the ML model learns the normal state of a given system, then an anomalous state fingerprinting methodology is proposed for state qualification, and to guide a tailored remediation action.

In [42], authors investigate how AI and edge computing can interwork. Often, AIML is used in edge network resource allocation problems that make surface at different layers and for different resources, such as CPU, radio and link resources. In [43], the authors conduct a comprehensive survey on the usage of AIML solutions for edge computing, focusing on deep learning models. Different degrees of integration between AI and cloud/edge computing are identified, going from fully cloudified environments where AI training and inference models run on the remote cloud, to an all-on device setting where the tasks are carried out on the device.

This coupling between AIML and networking is being facilitated by edge computing and network virtualization, standardization bodies are integrating AIML application requirements in system specifications. Namely, the NetWork Data Analytics Function (NWDAF) [44] has been proposed by 3rd Generation Partnership Project (3GPP) to support AIML in 5G core networks. However, various challenges are being discussed regarding different integration of training and inference sub-functions and the pipelining systems to get data to distributed AIFs.

2.3.2 Federated Learning Applications

A largely adopted strategy for geographically distributing AIFs down to network edges is Federated Learning (FL) [45]: it aims to prevent data collection aggregation at a central cloud, either for privacy issues or for latency constraints, or even both, by collaboratively training ML models at edge nodes. Two main steps are to be considered: (i) the local training of the ML model at the FL clients and (ii) the global aggregation of the updated parameters at the FL server. The FL process, if adequately configured and designed, can grant higher efficiency in terms of network bandwidth consumption and latency, besides increasing privacy thanks to data locality. The FL process itself can be repeated with several learning rounds until the model achieves a desired accuracy.

Mostly used for hand-held devices, FL is also being considered for in-network systems as well. In [46, 47], the authors propose NWDAF services based on FL; each 5G core NF can have its own NWDAF instance (NWDAF leaf) collecting data from its corresponding NF, training the ML model locally and aggregate parameters at a FL-server (root) NWDAF.

Many applications exist also for routing automation. In [48], the authors propose a federated learning architecture to optimize routing decisions in which SDN controllers cooperate to train an AIML model. The model takes as input both the network topology and the traffic state. The FL agents adapt the routing policy according to the predicted peak load and unutilized links.

Similarly, authors in [12] present how to use FL to distribute a centralized anomaly detection framework from [41]. The main goal is to cope with a set of challenges, mainly to scale with the increasing amounts of collected data and to reduce the training time for allowing a near-real time re-orchestration decision. Another work in [49] uses autoencoders-based multi-stackeholder recommender system for load prediction in cellular network, hence ensuring data privacy while offering the possibility to use 3rd-party services.

2.3.3 Stragglers control

A challenge in FL is to ensure clients are delivering meaningful data, i.e. data that can positively contribute to global learning mode, when the result of learning can change runtime inference tasks as related to anomaly/fault detection/prediction as in [12, 46, 47]. While this may not be important in arbitrary FL applications where corresponding inference results may not be expected to change from a training round to another, in-network applications can put stringent requirements on synchronous delivery of data to FL clients so as, for instance, to react to attacks or failures. Edge AIFs that are lagging beyond other edge AIFs in an FL setting are called 'stragglers' [50]. The main reason behind the appearance of stragglers is the combined heterogeneity of communication and computation delays. Stragglers slow down the learning process as the aggregation task at the FL server is only triggered once all the local parameters are received.

Two main approaches are proposed to handle stragglers in FL: careful client selection and adaptive learning model update.

2.3.3.1 Client selection/placement

In federated learning, performance degradation of the learning process is highly related to client selection. Several works from the literature consider optimizing both communication and computation latency. For instance, [51] and [52] consider dynamic client selection in hierarchical FL where both resource allocation and incentive mechanism are considered. The first step consists of an edge association task where each FL server offers rewards to FL participants to join its cluster. A second step considers choosing the model owner for each cluster. The authors consider the same processing capacity at the FL clients and allocate bandwidth resources (i.e. resource blocks) for a higher uplink bandwidth.

Similarly, in [53] the authors consider both system and data heterogeneity to minimize the occurrence of stragglers where the client selection is done for each round. In this work, the selected clients should have near-iid data where more bandwidth is allocated for clients with low computing capacity or poor channel conditions. In [54], the authors aim to find a trade-off between energy consumption and the number of active clients by choosing less clients during the first rounds. The authors propose an estimation of both computational and propagation latency while considering the

waiting time in the channel before sending the model parameters for aggregation.

In [55], authors show how FL clients selection can impact the global FL model quality and reduce training time, in a strategic game-theoretic setting to select FL participants based on the computing resources they offer: the goal is to achieve a given accuracy in the global model in an edge environment. A similar work is presented in [56], where a multidimensional procurement auction for FL clients selection is used to enhance model accuracy using a lower number of rounds.

Another technique to minimize stragglers is to increase the local training efficiency. In [57], the authors propose a FL policy to improve the training efficiency while considering heterogeneous clients. Clients with similar computational capacities are selected for training during a given round. Moreover, HWA can aslo be used to increase the learning efficiency. More precisely, edge computing provides AI with a convenient platform for models training and inferring, with a potential solution on accelerating computations on hardware [58]; HWA can be made available pervasively in edge networks, start from radio access and edge computing nodes. Besides reducing training and inference time depending on the type of accelerator [59], HWA can also decrease the energy footprint of AIML by up to 20 times [60, 61].

2.3.3.2 Adaptive global model update

Another way to control stragglers is to adapt the model update to stragglers occurrence. For instance, authors in [50] propose a straggler-robust scheme that adapts the node participation to the aggregation; the fastest subset of clients are selected to start the training, and the resulting parameters are then produced within a limited amount of time, and serve as a warm start for next rounds that also include the next fastest clients. In [62], the authors propose a live gradient compensation method to avoid stragglers for distributed learning tasks: only the gradient update for the k fastest workers is used, while combining the results from the slowest worker in the next iteration: the main goal is to reduce the overall training time while producing a near convergence error as fully synchronous gradient descent.

Enlarging the view to network communications, the authors in [63] consider both communication bottleneck and straggler delays in large scale distributed learning tasks: they combine a coding approach with a bandwidth sizing strategy to avoid bottleneck hence reducing stragglers. An enhancement of the gradient coding is proposed in [64], where the data is assigned to the edge AIFs in a distributed

manner so that a subset of model updates can be sufficient to compute the full gradient at the server side; then a dynamic clustering schema is associated to the set of edge AIFs, so that the completion time is improved. A hierarchical FL mechanism that encompasses both synchronous and asynchronous training schemes is proposed in [65] to mitigate straggling effect.

Edge computing provides AI with a convenient platform for models training and inferring, with a potential solution on accelerating computations on hardware [58]; HWA can be made available pervasively in edge networks, start from radio access and edge computing nodes. Besides reducing training and inference time depending on the type of accelerator [59], HWA can also decrease the energy footprint of AIML by up to 20 times [60, 61].

2.3.4 Summary

In our work, we aim at going beyond adaptive FL client section, while including the combined control of both network delay and training delay. In fact, we aim at compensating large deviations in data arrival from edge AIFs by the activation/disabling of HWA to reduce/increase the edge AIF training delay.

Many works in the literature investigated on the possible usage of HWA with AIML models. For example, in [66], authors motivate the use of Field-Programmable Gate Array (FPGA) to accelerate deep neural networks models where they have evaluated the reduction in the computation time while comparing it to a software implementation with different numbers of threads. The authors only consider the acceleration of inference as for the considered use-case training is done off-line. Note that in some cases, the training task should also be accelerated as the model needs to be updated. For instance, if we consider real-time anomaly detection, the new state of the system should be learned after a short period of time. Additionally, the authors in [67] explore different acceleration designs for a neural network-based model where both Graphics Processing Unit (GPU) and FPGA were considered. The authors inspected different configurations with the aim of identifying the optimal scenario for each acceleration approach. Nevertheless, the exploitation of HWA in distributed/federated learning client selection seems unexplored.

In our work, we propose an approach to control stragglers in in-network federated learning that combines both FL-AIF selection/placement and HWA enabling. We consider adaptive HWA usage to reduce the local training time at the FL clients with the goal to minimize variance in the end-to-end

combined network and training latency, as defined hereafter. Note that we do not address the combined usage of adaptive global model update, edge AIF placement and HWAs, left for future work.

artificial intelligence functions running in a federated learning setting at the edge of the network.

2.4 Conclusion

In this chapter, we have discussed several research issues related to MEC infrastructure deployment where both centralized and distributed deployments are considered. The combination of MEC and NFV is also discussed where we have shown that NFV paradigm is one of the proposed solutions for MEC application deployment. To shed light on resource orchestration at the application level, we have discussed the different challenges of the placement of MEC applications in distributed systems. In the next chapter, we present a first approach for optimizing MEC resource orchestration at the infrastructure level.

Chapter 3

Complexity vs Performance Tradeoff in MEC Infrastructure Orchestration

Contents

3.1 Introduction

Edge computing penetration in mobile access networks is the next barrier to break in communication networks. The virtualization of radio access functions currently under study is expected to trigger the deployment of edge cloud facilities in telecom operator PoP and CO, to serve the virtualization of both application servers and network functions. The problem of clustering network access points for their assignment to edge cloud facilities has been addressed in the literature. Nonetheless, the inclusion of KPIs such as robustness against traffic variations in the optimization process can increase its complexity excessively while hindering the achievable performance. Leveraging on a previous work in this area, in this chapter we explore how to reduce time and spatial complexity while introducing additionally a robust access point assignment target by using a spatial clustering pre-processing in the optimization problem, grouping together access points based on their spatio-temporal traffic profile.

Note that in this work, we consider ecosystems supporting non-ideal communication transport such as microwave [68] between the radio unit deployed at the base station level and the baseband unit that can be virtualized at the MEC host. In other words, we do not consider the ideal fronthaul connexions where the base station is statically connected to MEC hosts using optical fiber as in this case the association of BSs to MEC hosts is pre-established.

MEC orchestration algorithm scalability and result robustness are key concerns to address. The main contributions of this chapter are as follows[1]:

- We address the scalability-robustness challenge by extending a problem formulation and related algorithm in [34]. More precisely, we propose the integration of spatial clustering as a precomputation step to the algorithm in [34] to reduce the number of variables and constraints;
- We integrate in the spatial clustering optimization an objective that aims at making the access point to MEC host assignment more robust against traffic variations within a cluster of APs[2];
- We evaluate the proposed framework on a real-world dataset and We numerically show for which MEC network sizes the problem becomes tractable.

[1]This chapter was published in 2021 17th International Conference on the Design of Reliable Communication Networks [69].

[2]we use the terms AP and BS interchangeably.

3.2 Problem Statement

We describe our optimization framework as an orchestration problem that aims at assigning a group of BSs belonging to a given geographical area to a set of MEC hosts deployed at the edge network. The assignment operations come at a cost defined by the access latency for users connected to these BSs. On the other hand, unlike traditional Cloud datacenters, MEC hosts have limited capacities, thereby reallocating resources occasionally is requested to cope with traffic variation.

To reduce the spatial and temporal complexity of the orchestration process, we propose to group together BSs into clusters based on their spatio-temporal behavior so that the likelihood of traffic variation within the cluster is minimized. These requirements lead us to the adaptation of the orchestration model in [34] using the clusters in place of BSs, based on a robust assignment in the clustering process. To minimize the likelihood of cluster traffic variation, we opt for minimizing the variance of BS traffic volume within each cluster of BSs.

In Figure 3.1, we present an example of the assignment of base stations to MEC hosts.

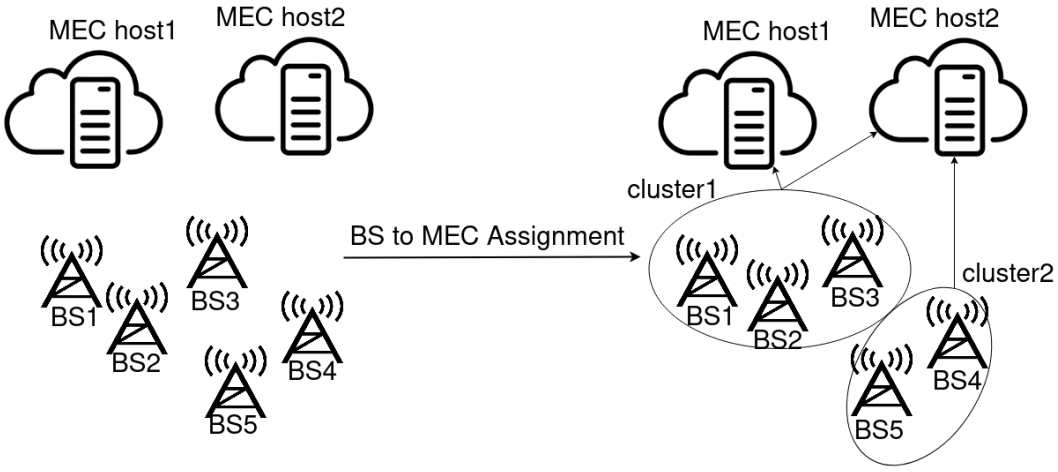


Figure 3.1: Example of BS-to-MEC assignment.

In Table 3.1 we define all notations used in the model.

3.3 Spatial Clustering Model

In our spatial clustering model, we search to group BSs so that for each time slot, the difference between their traffic demands is minimized. In order to have a linear and expressive robust clustering

Notation	Explanation
A	Set of all base stations (BSs).
K	Set of all MEC hosts.
T	Ordered set of time slots.
T'	$T' \subset T$ subset of T excluding the first time slot in T.
T''	$T'' \subset T$ subset of T excluding the last time slot in T.
C	Set of all clusters.
x_{ck}^t	Real variable, upper than 0 and less or equal to 1 if cluster c is assigned to MEC host k at time slot t, 0 otherwise.
y_{cjk}^t	Real variable, upper than 0 and less or equal to 1 if traffic demand of cluster c must be switched from MEC host j to MEC host k at time slot t, 0 otherwise.
M_c	Variable computing the maximum BS demand within cluster c.
m_c	Variable computing the minimum BS demand within cluster c.
d_c^t	Traffic demand of cluster c at time slot t.
ljk	Distance between the two MEC hosts j and k.
m_{ik}	Distance between the BS i and the MEC host k.

Table 3.1: MEC orchestration model notations.

objective, we express the traffic variance minimization by minimizing the gap between the maximum and minimum BS demand within the clusters.

For the instrumentation of the spatial cluster, we do as follows. We fix the number of clusters as corresponding to the number of MEC hosts. Given the collected traffic demands, we calculate the representative week by averaging demands of the same period of the week; we then aggregate the traffic demands of successive time periods aiming at reducing the number of intervals of time. In total we get a set of time slots that compose our training set. For the spatial clustering optimization we aim at grouping together BSs that have for each time slot t similar traffic demands so that the likelihood to have traffic fluctuation is reduced or can at least be relatively easy predictable - our tests to evaluate this assumption revealed to be extremely positive with the available dataset, which confirms that BS traffic profiles within a not too large time-slot do follow a similar temporal behavior over time [35].

We formulate the clustering task using an Integer Linear Programming (ILP). The mathematical formulation is as follows.

$$\min \sum_{c \in C} (M_c - m_c) \tag{3.1}$$

$$\text{s.t.} \quad \max_{t \in T} \sum_{i \in A} d_i^t x_i^c \leq Cap_{MEC} \qquad \forall c \in C \tag{3.2}$$

$$\sum_{c \in C} x_i^c = 1 \qquad \forall i \in A \tag{3.3}$$

$$\sum_{i \in A} d_i^t x_i^c \leq M_c \qquad \forall c \in C, t \in T \tag{3.4}$$

$$\sum_{i \in A} d_i^t x_i^c \geq m_c \qquad \forall c \in C, t \in T \tag{3.5}$$

$$\mathrm{x}_i^c \in \{0, 1\} \qquad \forall i \in A, c \in C \tag{3.6}$$

Where T refers to the set of all time slots, C is the set of clusters and A is the set of all BSs. d_i^t represents the traffic demand generated by the BS i at the time slot t. In our dataset, we have the traffic demands recorded for each BSs separately for each 10 minutes during a given period of time. To solve our problem we use the binary variable x_i^c, it is equal to 1 if the BS i belongs to cluster c, 0 otherwise. We also need to calculate the two real variables M_c and m_c where the former represents the demand traffic of a BS i representing the maximum for a time slot and belonging to cluster c, and the latter represents the demand traffic of a BS i representing the minimum for a time slot and belonging to cluster c, and finally Cap_{MEC} is the capacity of each MEC host. The objective function in (3.1) aims at minimizing the difference, for all the clusters, between the maximum and minimum traffic demands yield by BSs belonging to the same cluster. Constraint (3.2) ensures that the maximum traffic demands that can be handled by each cluster must not exceed MEC hosts capacity. In (3.3) we guarantee that a BS belongs to exactly one cluster. (3.4) and (3.5) ensure the M_c and m_c computation, i.e., the maximum and the minimum traffic demand generated by a BS that belongs to cluster c at time slot t, respectively. (3.6) is an integrality constraint.

3.4 MEC Orchestration Model

Our proposal consists of solving the orchestration problem where we apply the same orchestration decision on BSs belonging to the same cluster. For this purpose we extended the orchestration model from [34] to fit with our spatial clustering model. The goal of the orchestration model is to assign a group of APs belonging to a given geographic area, to a set of MEC hosts. We consider the possibility

of having a cluster composed of only one AP, in this case, the model represents a single AP assignment problem, which refers to the baseline algorithm.

Let us consider a user equipment connected to an AP; the assignment operation of its traffic to a given MEC host yields a cost defined by the access latency. We assume that hosting demands of a given AP on a MEC host consists of allocating one VM for each UE. On the other hand, and unlike traditional datacenters, MEC hosts have limited capacity, thus switching AP demands from a MEC host to another is sometimes requested in order to cope with traffic variation. Given the lower traffic granularity at MEC hosts, switching operations entails a cost for operators because it could generate service-level-agreement violations and hence a VM workload variation across MEC hosts to get back to nominal conditions.

The model is represented by equations from (3.7) to (3.13) (see notations in Table 3.1). The objective formulated in (3.7) aims to find an assignment plan for each cluster of BSs to the set of MEC hosts for each period of time, where each cluster can be composed of one or multiple APs. We aim to minimize both assignment and deployment costs. In (3.8) we ensure that the overall demands assigned to a MEC host must not exceed its capacity. Constraints (3.9), (3.12) and (3.13) give the possibility to assign a cluster of APs to one or more MEC hosts for each time slot. In fact, in this case, the AP demands can be split and assigned to different MEC hosts. If we have the nearest MEC host with a very small available capacity, the proposed solution allows us to assign the remaining demands to other MEC hosts.

$$\min \sum_{t\in T}\sum_{c\in C}\sum_{\substack{(j,k)\in \\ K\times K}} d_c^t l_{jk} y_{cjk}^t + \sum_{t\in T}\sum_{c\in C}\sum_{k\in K} d_c^t m_{ck} x_{ck}^t \tag{3.7}$$

$$\text{s.t.} \sum_{c\in C} d_c^t x_{ck}^t \leq Cap_{MEC} \qquad \forall k \in K, \forall t \in T \tag{3.8}$$

$$\sum_{k\in K} x_{ck}^t = 1 \qquad \forall c \in C, \forall t \in T \tag{3.9}$$

$$x_{ck}^t = \sum_{j\in K} y_{cjk}^t \qquad \forall c \in C, \forall k \in K, \forall t \in T' \tag{3.10}$$

$$x_{ck}^t = \sum_{j\in K} y_{cjk}^{t+1} \qquad \forall c \in C, \forall k \in K, \forall t \in T'' \tag{3.11}$$

$$x_{ck}^t \in [0,1] \qquad \forall c \in C, \forall k \in K, \forall t \in T \tag{3.12}$$

$$y_{ckj}^t \in [0,1] \qquad \forall c \in C, \forall j,k \in K, \forall t \in T \tag{3.13}$$

Constraints (3.10) and (3.11) ensure the balancing of demand flows for each cluster, each MEC

host and each time slot. More precisely, the right-hand side of (3.10) represents the overall fraction of demands of APs belonging to cluster c *incoming* to MEC host k at time t, possibly being switched from other MEC hosts; this needs to be consistent with the value of the corresponding x^t_{ck} variable. Similarly, the right-hand-side of each (3.11) represents the overall fraction of demands of APs belonging to cluster c *outgoing* from MEC host k at time t, possibly being switched to other MEC hosts. Since the left-hand-sides are identical, (3.10) and (3.11) impose incoming and outgoing demand fractions to be equal.

3.5 Experimental Setup

3.5.1 Deployment of the MEC orchestration process

Following the algorithm process depicted in Figure 3.2, we evaluate the MEC orchestration framework proposed in the previous section. We first run the clustering model at the preprocessing phase then we apply the MEC orchestration model on the resulting clusters. Two types of inputs are fed to the clustering model, (i) time series data; refers to the user traffic demands aggregated at the BSs; and (ii) configuration parameters such as BS positions. The orchestration model is then applied on the clusters to provide the assignment and switching plan for a set of time slots. Note that as a by-product, an evaluation of the total costs and some other statistics is possible.

3.5.2 Datasets

We used a real traffic dataset from a national mobile network made available in the frame of the French ANR CANCAN (Content and Context based Adaptation in Mobile Networks) project. The dataset gives us access points downlink volume information every 10 minutes for a period of three months in 2019, for Paris and Lyon metropolitan area networks. The collection process takes into consideration both 3G and 4G connections and records data on a per-user basis that are aggregated at the AP level. Paris dataset contains a larger amount of demands when compared to Lyon dataset. In fact, we have chosen 1908 base stations for the area of Paris and 332 for the area of Lyon [3]. We split our datasets into two parts: we used the first two-thirds to train both the clustering and orchestration models, and the remaining third, i.e., held-out data, to evaluate the quality of the solution.

[3]The content of the dataset is private, additional details cannot be provided.

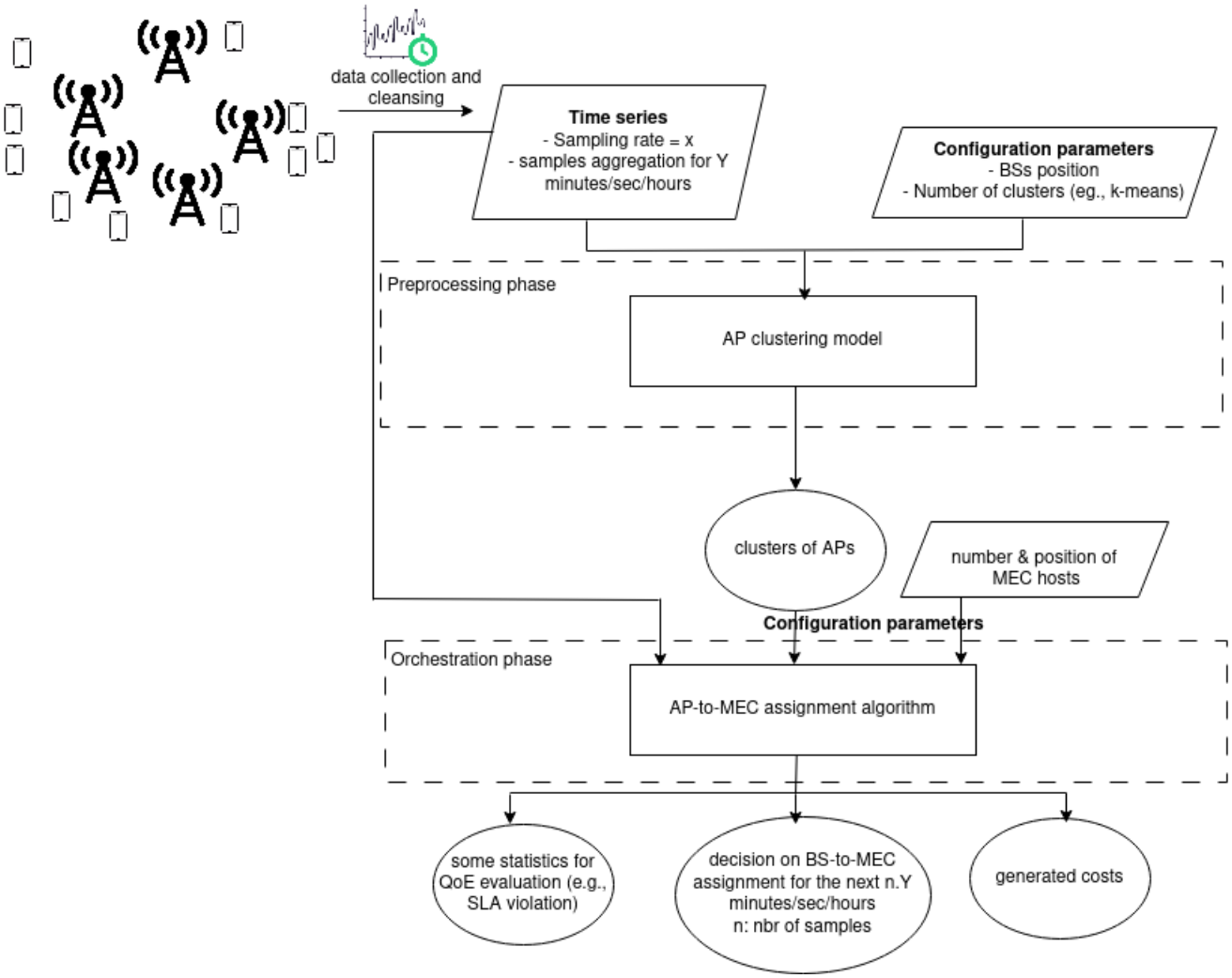


Figure 3.2: MEC orchestration framework - Global view.

We implemented our model in A Mathematical Programming Language (AMPL) [70] using CPLEX as the linear solver [71]. We run our algorithms on an Ubuntu Server 14.04 LTS virtual machine with 64 GB of RAM and 8 $\times$ 2.5 GHz CPU cores.

3.5.3 Framework evaluation

We assess the results through Cumulative Distribution Function (CDF) of spatial and time complexity metrics, i.e., maximum memory usage (in GB) and execution time (in seconds), as well as the optimality gap (%) and the assignment and switching costs. We use up to 50 different locations for MEC hosts, hence using different numbers of MEC host facilities to generate the training set, for the cities of Paris and Lyon. MEC locations are generated using a centroid based clustering, i.e., K-means clustering where the centroids of BS clusters represent MEC hosts locations. For each simulation we randomly generate the parameter representing the number of time the k-means algorithm is executed, then the best results are returned based on inertia. We evaluate the following four algorithms to solve the orchestration problem:

- 'MECA': solving the reference orchestration model without spatial clustering, i.e., (3.7)-(3.13) with $C \equiv A$;
- 'MECA-CS': solving the reference orchestration model with spatial clustering, i.e., (3.1)-(3.13);
- 'MECA-CG': solving the reference orchestration model without spatial clustering and using the dynamic variable generation approach proposed in [34];
- 'MECA-CG-CS': as MECA-CG but with spatial clustering precomputation.

In the following, we present simulation results generated by both Lyon and Paris datasets, using different MEC infrastructure sizes: 10, 20 and 30 MEC facilities, and 20 and 50 MEC facilities respectively. Due to its high memory consumption, we could not execute the MECA approach for the two highest MEC infrastructure sizes, i.e., 50 and 30, for Paris and Lyon, respectively.
MECA-CG-CS was the least memory consuming case on single computations. Contrary to expectations, it increased on average by 4.9 GB (1600%) for Paris dataset using 20 MEC hosts and by 1.3 GB (540%), 2.7 GB (700%) and 4.9 GB (980%) for Lyon dataset using respectively 10, 20 and 30 MEC

hosts when post-processing the intermediate solutions to retrieve the variable vectors. The maximum execution time limit is set to 17000 s for all instances, seldom reached.

3.6 Performance Evaluation

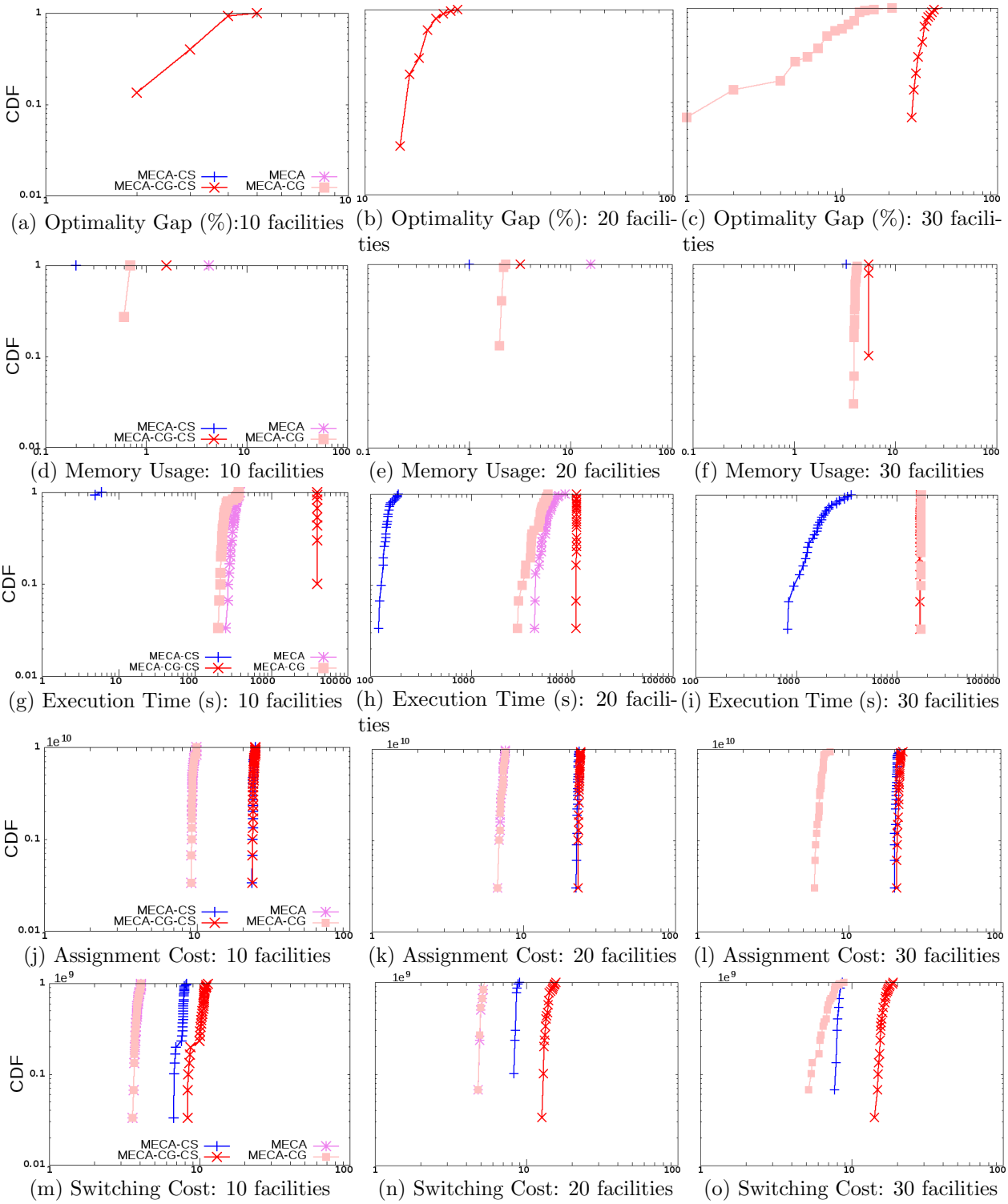
3.6.1 Evaluation using Lyon dataset

Figure 3.3 depicts the numerical results when using 3 different sizes for our infrastructure (10, 20 and 30 facilities) and traffic demands from Lyon dataset. We report the results of the 5 aforementioned metrics.

Figures 3.3a, 3.3b and 3.3c present the distribution of the optimality gap values using Lyon dataset and the three different sizes for the MEC infrastructure. We note that: 0% optimality gap was reached by MECA-CS, MECA and MECA-CG after a finite execution time when using 10 and 20 MEC hosts, contrary to MECA-CG-CS that reached optimality gaps between 2% and 5% and between 10% and 17% respectively, depending on the MEC hosts locations. This can be explained by the use of large scales, i.e., aggregated demands of all BSs that belong to the same cluster to get the cluster traffic demand. For 30 MEC hosts, we reached the 0% optimality gap only with MECA-CS in a finite time. For MECA-CG, values are between 0% and 20%. Meanwhile, MECA-CG-CS has the highest values that varies between 28% and 44%.

Figures 3.3d (resp 3.3e and 3.3f) depicts the distribution of the maximum amount of memory used by the processes run by the proposed approaches in GigaBytes (GB) when using 10 MEC hosts (resp. 20 and 30 MEC hosts). We note that: for all cardinalities, the best case always corresponds to MECA-CS with a constant memory consumption for all the 30 MEC hosts locations, i.e. 0.15 GB for 10, 1 GB for 20 and 3.29 GB for 30 MEC hosts, followed by MECA-CG with maximum consumption of 0.72 GB for 10 MEC hosts, more than 2 GB for 20 and 4.19 GB for 30. MECA has the highest memory consumption peak for both cases 10 and 20 where it reaches respectively 4 GB and 15.7 GB. However, when using 30 MEC hosts simulations has stopped before reaching any solution due to its high memory consumption. Meanwhile, MECA-CG-CS has an intermediate consumption between MECA and MECA-CG for all cardinalities, i.e. 1.5 GB, 3.2 GB and 5.4 GB.

In Figures 3.3g, 3.3h and 3.3i we present the CDF histograms of the required execution times by the proposed approaches and for the 30 MEC hosts different locations for each of the three infrastructure



(a) Optimality Gap (%):10 facilities (b) Optimality Gap (%): 20 facilities (c) Optimality Gap (%): 30 facilities

(d) Memory Usage: 10 facilities (e) Memory Usage: 20 facilities (f) Memory Usage: 30 facilities

(g) Execution Time (s): 10 facilities (h) Execution Time (s): 20 facilities (i) Execution Time (s): 30 facilities

(j) Assignment Cost: 10 facilities (k) Assignment Cost: 20 facilities (l) Assignment Cost: 30 facilities

(m) Switching Cost: 10 facilities (n) Switching Cost: 20 facilities (o) Switching Cost: 30 facilities

Figure 3.3: Cumulative distribution function of the evaluated metrics: execution time (s), memory usage (GB), optimality gap (%) and assignment and switching costs - Lyon dataset.

sizes. We note that: comparing the two approaches MECA-CS and MECA highlights the contribution of the spatial clustering: MECA-CS is the fastest approach with an execution time less than 7 s, between 120 s and 190 s and less than one hour for the three cardinalities. Nevertheless, MECA reaches 390 s and 2 hours of execution time for the first two infrastructure sizes. For 30 MEC hosts the algorithm did not get any results. For 10 and 20 MEC hosts, MECA-CG requires between 200 s and 380 s and 1 hour and a half whereas MECA-CG-CS gives the worst case with an execution time exceeding 1 hour and 3 hours respectively. Hence, both MECA-CG and MECA-CG-CS reached the execution time limit which is 17000 s when using 30 MEC hosts.

We present in Figures 3.3j, 3.3k and 3.3l the distribution of the assignment costs values yield by the proposed approaches. We note that: the assignment costs yield by approaches using spatial clustering model are higher than costs generated by the approaches spatial clustering-free. For 10 and 20 MEC hosts cases a 0% optimality gap was reached by both MECA and MECA-CG, so the two assignment costs are equal. We can also notice that the assignment cost decreases when we broaden the MEC infrastructure size.

We present in Figures 3.3m, 3.3n and 3.3o the distribution of the switching costs values yield by the proposed approaches when using Lyon dataset. We note that: for 10 and 20 MEC hosts (3.3m and 3.3n), the switching costs are lower when not using the spatial clustering as explained before. MECA and MECA-CG have the same and lowest switching cost, followed by MECA-CS (less than double) and finally MECA-CG-CS. On the other hand, when using 30 MEC hosts (3.3o) we notice that MECA-CS and MECA-CG have the same switching cost for some MEC hosts locations. MECA-CS and MECA-CG-CS have different switching costs because this latter could not reach the 0% optimality gap. The switching cost increases when increasing the MEC infrastructure size from 10 to 20 facilities. However, increasing it from 20 to 30 has decreased the switching cost for MECA-CS and increased it for MECA-CG.

3.6.2 Evaluation using Paris dataset

The 0% optimality gap was reached for all the approaches and for all cardinalities, except for MECA when using 50 MEC hosts. As aforementioned, it stopped before getting any results.

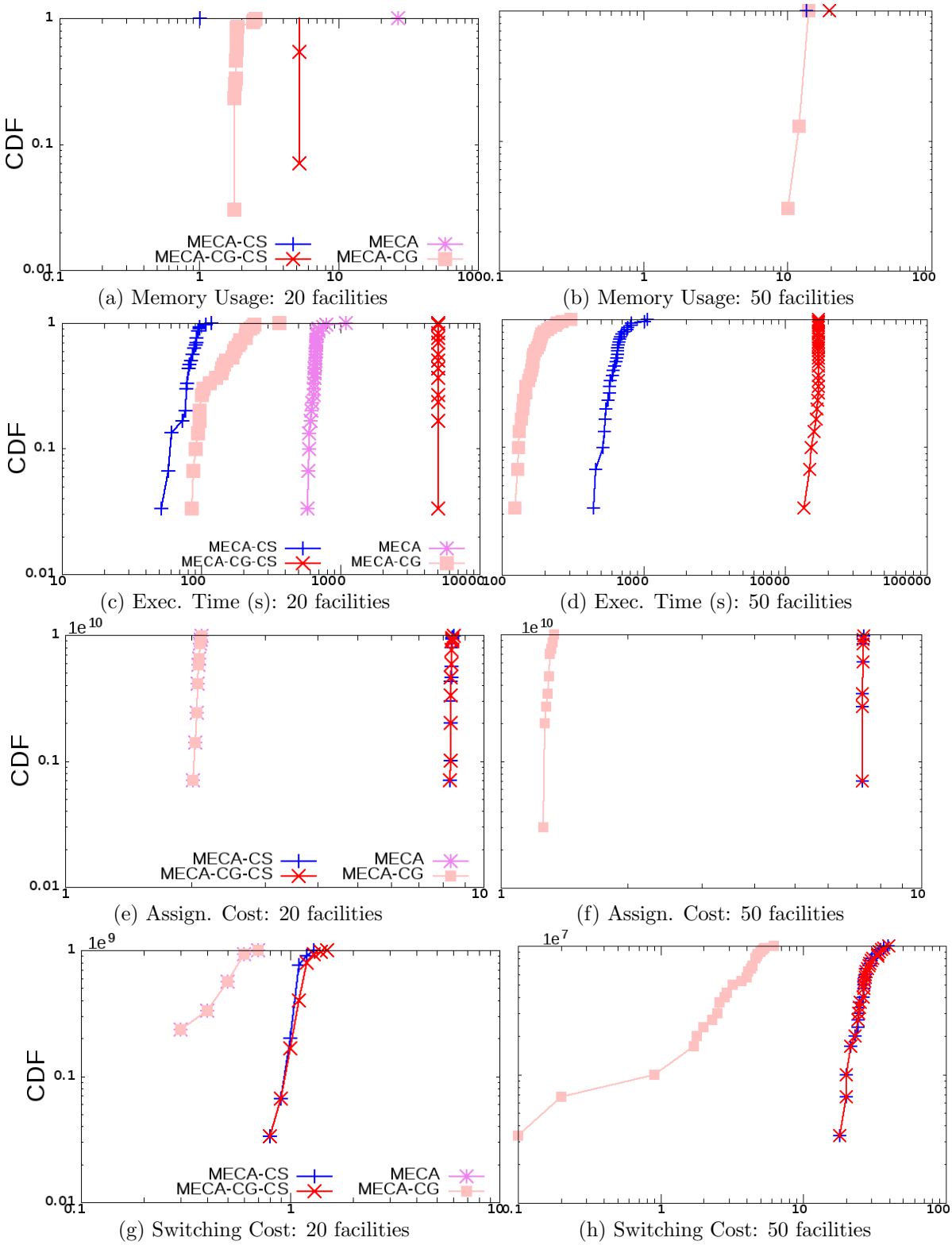
In Figure 3.4 we present the distribution of the maximum memory usage in GigaBytes (GB), the execution t3ime in seconds (s) and the assignment and switching costs for Paris dataset for two

different sizes for the MEC infrastructure, i.e, 20 and 50 facilities.

The distribution of the maximum memory used by each of the approaches is depicted in Figures 3.4a and 3.4b using Paris dataset and respectively 20 and 50 MEC hosts. We notice that: when using 20 MEC hosts, MECA-CS has a constant maximum memory usage through all the proposed MEC hosts locations and it represents the lowest value (1 GB) compared to all the other approaches, followed by MECA-CG and MECA-CG-CS (2.5 and 5.25 GB as maximum values respectively). Meanwhile, MECA is the most memory consuming and it has also a constant consumption through all the proposed locations (26 GB). For 50 MEC hosts, MECA-CS and MECA-CG have a close maximum memory usage on average, the former has a constant consumption equal to 13.6 GB while the latter consumption varies between 10 GB and 14 GB. However, MECA-CG-CS has the highest values that reach 19.4 GB. MECA has the highest memory consumption and it stopped before reaching the final solution because of lack of memory. Increasing MEC hosts number from 20 to 50 has increased the maximum amount of memory used by each of the proposed approaches.

Figure 3.4c (resp. Figure 3.4d) represents the distribution of execution time values required by each approach when using 20 MEC hosts (resp. 50). We note that: MECA-CS is the fastest approach when using 20 MEC hosts followed by MECA-CG with execution time values that go from 50 s to 120 s and from 86 s to 360 s respectively. On the other side, when increasing the infrastructure size to 50 MEC hosts MECA-CG becomes the fastest one reaching 300 s, followed by MECA-CS where the execution time is between 440 s and 1000 s. MECA needs a higher execution time that reaches 600 s at least and 1000 s at most for 20 MEC hosts. However, MECA-CG-CS represents the highest execution time and requires around 5000 s for 20 MEC hosts and reaches the execution time limit with 50 MEC hosts. It is worth mentioning that it was clearly stated in [18] that 20 MEC hosts is sufficient to satisfy strict requirements in terms of latency and bandwidth.

The distribution of the assignment costs is presented in Figures 3.4e and 3.4f for both 20 and 50 facilities, we can notice that: the approaches without the spatial clustering yield a lower assignment cost compared to the ones using it. Let us underline that the spatial clustering adds a constraint to the orchestration problem that produces the same assignment plan to all BSs that belong to the same cluster. When the 0% optimality gap is reached, MECA-CS and MECA-CG-CS (MECA and MECA-CG respectively) have roughly the same assignment cost: a little difference can be noticed due to numeric precision used by the two methods.



(a) Memory Usage: 20 facilities

(b) Memory Usage: 50 facilities

(c) Exec. Time (s): 20 facilities

(d) Exec. Time (s): 50 facilities

(e) Assign. Cost: 20 facilities

(f) Assign. Cost: 50 facilities

(g) Switching Cost: 20 facilities

(h) Switching Cost: 50 facilities

Figure 3.4: Cumulative distributed function of the evaluated metrics: execution time (s), memory usage (GB) and assignment and switching costs - Paris dataset.

In Figures 3.4g and 3.4h, we present the distribution of the switching costs. We notice that: there is a slight cost difference between the two approaches (with vs without spatial clustering precomputing). As explained, the approaches spatial clustering-based produce additional costs due to proposing the same switching plan to BSs that belong to the same cluster. Achieving the 0% optimality gap produces the same switching costs for MECA-CS and MECA-CG-CS (and for MECA and MECA-CG respectively), and increasing the number of MEC hosts has reduced both the switching and the assignment costs for all the approaches.

3.7 Conclusion

In this chapter, we focused on the optimization of algorithms that deal with base-station access-point to MEC hosts assignment orchestration decisions by taking into account an assignment objective robust against traffic fluctuations. For this purpose, we proposed a spatial clustering model which consists of grouping together base-station access points into clusters that reveal the same spatio-temporal traffic through time. Afterwards, a data-driven solution for MEC orchestration was added to the model. The results from extensive simulation on a real world dataset show that our approach outperforms existing algorithms while helping reduce time and space complexity especially for small to medium instances, i.e., 10, 20 and 30 MEC hosts for Lyon city and 20 MEC hosts for Paris city. As aforementioned, a previous work has evidently demonstrated that using around 20 MEC hosts for the region of Paris would therefore be more than sufficient for realistic massive MEC service deployment, even with strict constraints on latency and maximum link utilization.

Despite the fact that the spatial clustering model entails an additional cost due to the constraint that imposes the same assignment and switching plan for base-station access points belonging to the same cluster, numerical results have shown that our framework can be carried out in a near-real-time manner. In the next chapter, we extend the pre-processing phase with the aim of improving both user and switching costs while including additional parameters to evaluate the degree of controlling MEC hosts capacity.

Chapter 4

Pairwise Access Point Clustering for MEC Infrastructure Orchestration

Contents

4.1 Introduction

In this chapter, we extend the framework presented in Chapter 3 to include a more variate set of clustering fitness functions, comparing them in terms of reliability. Reliability is hereafter meant as the capacity of not exceeding the allocated computing capacity under varying traffic loads. We demonstrate how, thanks to the preliminary spatial clustering, we can integrate different orchestration flavors to make the MEC orchestration decision framework more robust against traffic fluctuations, while taking into consideration secondary performance indicators. We introduce a set of clustering models to be deployed at the pre-provisioning phase. We go through extensive simulations on real-world traffic demands to evaluate the performance of the proposed solutions. In addition, we show how MEC hosts capacity violation can be decreased when integrating access points clustering into the orchestration model, by investigating on solution accuracy when applied on held-out users traffic demands. The obtained results show that our approach outperforms two state-of-the-art algorithms, reducing both memory usage and execution time, by 46% and 50%, respectively, in comparison to a baseline algorithm. It surpasses the two methods in gaining control over MEC hosts capacity usage for different maximum achieved occupancy levels on MEC hosts.

Scalability and robustness are major challenges that arise when dealing with MEC resource orchestration problems [72]. In this chapter, we shed light on the scalability-robustness challenge by extending the MEC orchestration framework in Chapter 3. We propose a portfolio of AP clustering algorithms, to integrate as a preprocessing phase to the actual orchestration problem, scheduling AP-to-MEC facility assignments over time. Our clustering algorithms are meant to show the flexibility we can benefit from MEC orchestration, while evaluating the impact in terms of robustness when considering heterogeneous demand profiles.

The main contributions of this chapter are as follows[1]:

- We formulate a collection of AP clustering algorithms that we integrate into a MEC orchestration baseline algorithm [34], with the aim of reducing both execution time and memory usage, while integrating robustness criteria in the orchestration problem;
- We train the proposed frameworks using a real-world dataset, composed of demands collected at

[1]This chapter was published in *IEEE Transactions on Network and Service Management, vol. 19, no. 3, pp. 2738-2750, Sept. 2022* [73].

two different regions in France;

- We apply the resulting assignment plans on held-out data, in order to assess how well the proposed assignment plans can adapt to access point changing demands by evaluating the violation of servers capacity;
- We finally compare our algorithms to two different approaches from the state-of-the-art. Numerical results show that our algorithms outperform these existing solutions in terms of robustness and computing performance.

4.2 Problem Statement

The MEC orchestration problem is decomposed into two independent but connected phases: a pre-processing phase that consists of grouping together APs into clusters based on some criteria, and a second phase that holds the assignment of the resulting clusters to the MEC hosts with available capacity. The assignment task takes into consideration the distance between access point to MEC hosts to which it is assigned to, thus representing the user costs, also called the assignment cost in the rest of the chapter. We propose to trigger assignment operations for each time period, where the duration of a time period can go from few seconds up to few hours. Since MEC hosts capacity is limited, and due to users traffic variations, VM resource migration can be required; these operations yield a deployment cost of the network, referred to as switching cost. The AP-to-MEC assignment problem is presented in Section 3.4. In the following, we introduce the pre-proccessing phase with the clustering approaches. Table 4.1 lists the notations used.

The proposed framework is based on a training process that consists of using historical data i.e., access point traffic demands, to identify the parameters of a model. We define two different models, one model that groups together APs based on a given criterion at the clustering phase, and a second one that assigns the resulting clusters to the available MEC hosts at the orchestration phase.

4.3 Multi-objective Access Points Clustering

The idea of performing access points spatial clustering as a preprocessing to the optimization problem is to, from the one hand, take more robust decisions with respect to traffic variations by

A	Set of all access points.
$T = \{1, \ldots, \tau\}$	Ordered set of τ time slots.
${d_i}^t$	Parameter, represents the traffic demands of AP i at time slot t.
${d_c}^t$	Parameter, represents the demands of the cluster c at time slot t.
$d_i'^t$	Parameter, represents the variance of demands, of the same period of each week for AP i (for example, if the number of samples of the dataset corresponds to w weeks, d'^t_i is then, the variance of all demands of AP i at the same time period t over all the w weeks).
$\bar{d_i}$	Parameter, represents the average of traffic demands of access point i through all the time slots.
c_{ij}	Parameter, identifier of the pairwise clustering criterion.
Q	Parameter, represents the capacity of each MEC host. All MEC hosts have the same capacity.
δ_{ij}	Parameter, represents the (complete link) distance between the two AP clusters i and j;
$\tilde{\delta}$	Parameter, represents the maximum distance between each couple of AP that belongs to the same cluster.
z_{ij}	Binary variable, takes value of 1 if APs cluster i is paired to APs cluster j to form a cluster, 0 otherwise.

Table 4.1: Notations.

grouping together BSs that satisfy a given performance target and, from the other hand, decrease both execution time and memory space of the assignment problem (3.7)-(3.13) thanks to variable aggregation and constraints reduction. We propose an extension of the spatial clustering model proposed in section 3.3. In order to ease reaching optimal configurations, we set an iterative pairwise access point clustering instead of grouping an indeterminate number of APs together.

The motivation is to use simpler combinatorial models by merging two APs as an AP pair, based on a different set of criteria. Depending on the scales of the problem (i.e., number of APs), this pairwise clustering can be iterated so as to group, at a second stage, two pairs of APs within a cluster; as so on so forth, if needed, further hierarchical clustering can happen. We then apply the orchestration decision on the set of resulting clusters. The clustering criterion is meant to allow granting a robustness flavor to the orchestration model, namely in terms of robustness against load variation in time. The goal is to reduce MEC host capacity violations.

The generic mathematical formulation of the clustering approach is as follows:

$$\min \text{ or } \max \sum_{i \in \bar{A}, j \in \bar{A}} c_{ij} * z_{ij} \tag{4.1}$$

$$\text{s.t.} \sum_{j \in \bar{A}} z_{ij} + z_{ji} = 1 \qquad \forall i \in \bar{A} \tag{4.2}$$

$$z_{ij} = 0 \qquad \forall i, j \in \bar{A} : \delta_{ij} > \tilde{\delta} \tag{4.3}$$

$$z_{ij} \in \{0, 1\} \qquad \forall i, j \in \bar{A}$$

The set $\bar{A}$ contains one element for each AP cluster. The objective function in (4.1) aims at minimizing (maximizing, respectively) the clustering cost value expressed using the criterion parameter c_{ij} which defines the degree of similarity or difference according to which elements (access points or clusters of access points) are grouped in the same cluster. More precisely, it is given by the sum of costs for those pairs of AP clusters which are joined.

Constraints (4.2) ensure that each cluster $i \in \bar{A}$ is paired to exactly one other cluster (either i is paired to some j, or j exists, which is paired to i). Constraints (4.3) ensure that such a pairing is made only among AP clusters whose distance does not exceed a given threshold. Note that having z_{ij} =1 implies the creation of a cluster that merges APs (resp. groups of APs) i and j together, in which case all other z variables involving i are set to 0, to technically keep consistency (i.e. all nodes with degree one) on the directed graph model we employ.

Initially, $\bar{A} = A$. That is, single APs are paired. Then, single elements of $\bar{A}$ are replaced by the pairwise clusters which are formed. After all replacements are made, new cluster criterion parameters and distances are computed for the elements of $\bar{A}$, and the clustering process is iterated. For each iteration, we compute the distance between each couple of APs that belongs to different clusters. Only clusters with distances that are lower than the threshold $\tilde{\delta}$ are grouped together. We propose two classes of criteria as in the following.

4.3.1 Clustering-based on load differences

This class of criteria aims at grouping together access points depending on the demand differences without taking into consideration AP demand profiles. We propose four different criteria:

- MIN-MAX. To reduce the absolute value of the difference in demands for each couple of APs during all the time periods. The goal is to obtain clusters with similar demands for each period of time. In fact, this criterion represents an enhanced version of the single-one used in Section 3.3 (Chapter 3).

- MIN-SUM. To minimize the average of differences in demands for all the time periods, for each couple of APs belonging to the same cluster. The goal is to group together APs where the average of their demands is minimized through all time slots.

- MAX-MAX. To group each couple of access points where the difference between their demands for each time slot is maximized. This criterion yields clusters of APs with highly different traffic demands.

- MAX-SUM. To maximize the average of demand differences of each couple of APs and for all the time periods in order to produce clusters of APs that behave differently through time.

4.3.2 Clustering-based on load correlation

This class of criteria uses different forms of correlation among AP load, taking into consideration the AP demand profiles. We propose four different criteria:

- MIN-CORR. To find negatively correlated couples of APs to group them together into the same cluster. To do so, we propose an expression that defines the relationship between demands of each couple of APs expressed as a ratio. Minimizing this ratio leads to a maximization of the difference in APs demand profiles.

- MAX-CORR. To search for couples of APs where the correlation between their demands is maximized through time in order to have positively correlated demands in the same cluster. In this way, the resulting clusters group together APs with highly similar demand profiles.

- MIN-CORR-VAR. To have couples of APs with negatively correlated variance of demands. In order to achieve this objective, we minimize the correlation of the variance of demands of APs i and j. An access point with high demand variance will be then merged with an AP having low demand variance.

- MAX-CORR-VAR. To maximize the correlation of demand variance between couples of APs belonging to the same cluster, with a view to have each couple of AP with demand variance that are positively correlated grouped into the same cluster. Thus, an AP having high variance in its demands through time will be merged with an AP of the same demand profile.

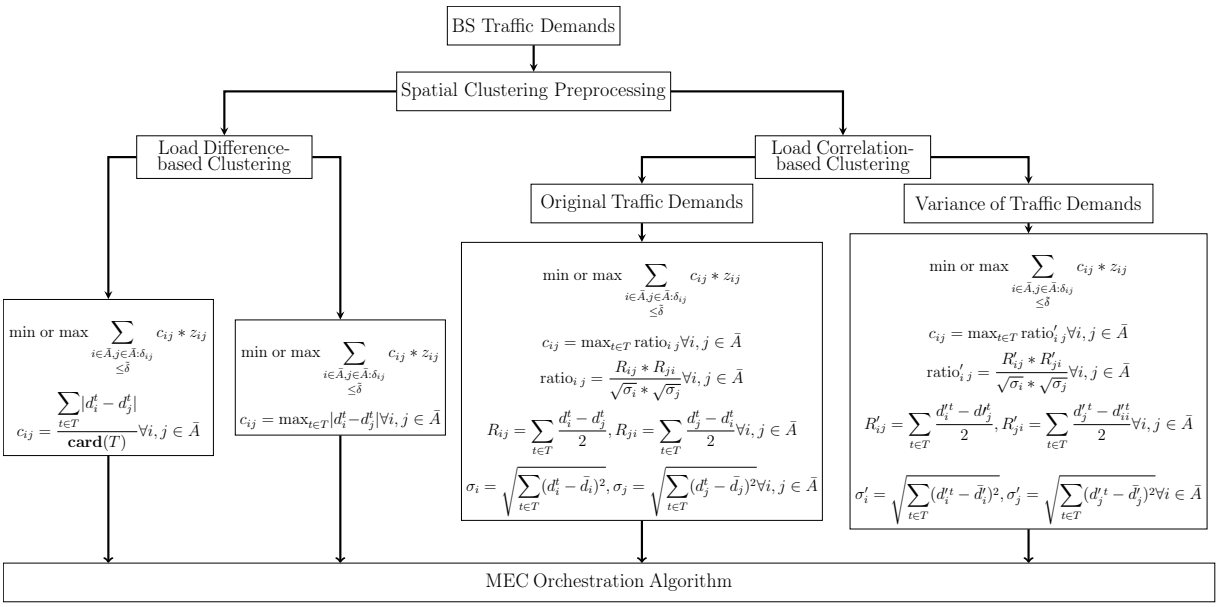


Figure 4.1: MEC Orchestration Framework Options.

We embed in Figure 4.1 the mathematical expression for each criterion. The complete mathematical formulations are given in the appendix A.

4.4 Experimental Results

In the following, we first describe the parameters used to evaluate our framework, then we provide a numerical evaluation of the MEC orchestration approaches using different metrics while comparing them to two approaches from the state-of-the-art.

We solve the resource orchestration problem using the eight clustering variants in Section 4.3 and the following additional algorithms:

- 'MECA': solving the reference orchestration model without spatial clustering, i.e., (3.7) to (3.13), this refers to the baseline MEC assignment algorithm, which we refer to in the following as the benchmark;

- 'HYPERBOLIC': solving the reference orchestration model with the HYPERBOLIC-KMEANS spatial clustering at the state-of-the-art [36];

We also refer to 'MECA-CS^{+}' as solving the reference orchestration model with MIN-MAX spatial clustering in the pre-processing phase (enhancement of the MECA-CS algorithm in Section 3.3);

4.4.1 Dataset and parameter setting

We used a dataset with traffic demands collected at the core and access network of a French mobile operator, at a national scale and for a period of three months. More details about the dataset and the system characteristics are available in Section 3.5.2.

For the clustering algorithms described in Section 4.3, we perform (i) for Lyon dataset, only once the pairwise clustering, so clusters of two APs are formed, while (ii) for Paris dataset, we perform two pairwise clustering iterations, to decrease the memory usage and execution time otherwise too high given the higher number of APs. Indeed, the number of hierarchical pairwise clustering iterations can be customized based on the trade-off between execution time and assignment cost, as discussed hereafter.

For the simulations, we generate MEC facilities locations using a variant of the K-means clustering method, called weighted K-means, such that each MEC host location is the centroid of a given group of APs; the weight is represented using APs demands dispersion. In this way, MEC hosts positions are generated depending on the access points demands profiles. We fix the number of MEC facilities to 20 servers for both Lyon and Paris datasets. Then, we randomly generate 10 different configurations for MEC hosts locations with the aim of producing different inputs to train our MEC orchestration algorithm.

4.4.2 Numerical evaluation

We use for this evaluation Lyon dataset and we analyze the quality of the solution based on the following metrics:

- the memory usage and execution time: represented by the execution time (s) and maximum memory usage (GB);
- the assignment and switching costs;

- the total cost gap convergence in percentage, against the benchmark;
- the distribution of BS-to-MEC distances during the period of the training (refers to a representative week) in kilometers;
- the number of switching operations.

In Table 4.2, we summarize the average ± standard deviation of the aforementioned metrics results.

Figures 4.2 to 4.6 depicts the distributions for each of the aforementioned metrics. We now draw our observations on these results as follows.

4.4.2.1 Execution time

Figure 4.2 reports the distribution of execution times experienced with each case. We can remark that the slowest algorithm is the benchmark (MECA) solution where the average execution time exceeds half an hour. In fact, applying the clustering algorithm in the preprocessing, as done for the other cases, helps reduce the execution time as proven in Chapter 3.

We can also notice that the MAX-CORR has the lowest execution time among the eight proposed algorithms, followed by MECA-CS^{+}. The highest execution times are yield by the algorithms based on load difference which are MAX-MAX and MAX-SUM, respectively. The HYPERBOLIC clustering is globally the fastest one with an average of 376 s; due to the fact that the number of APs per cluster is not fixed, which generates clusters with a higher number of APs compared to the other models, hence reducing both the memory usage and execution time, as shown in Chapter 3. However, this gain in the MEC-to-AP assignment phase comes at the expense of a much longer pre-processing phase time as shown in Table 4.2. This time, moreover, increases with the dataset size: tests with the Paris dataset show that HYPERBOLIC clustering needs more than 24 hours to be trained versus few dozen of seconds for the other cases performing iterative pairwise clustering.

[2]The maximum memory usage metric was the same for each of the MEC hosts configurations, because we do not change the data size for each of the proposed configurations.

Models	Execution Time (s)		Memory $(GB)^2$	Assign. Cost e^{+10}	Switch. Cost e^{+9}	Gap
	Orchestration	Clustering				
MECA-CS$^+$	753.84 ±160.50	0.902 ±0.034	7.99	7.79 ±0.169	3.74 ±0.53	1.31 ±0.02
MIN-SUM	838.44 ±208.75	1.727 ±0.049	7.99	7.65 ±0.139	3.69 ±0.58	1.29 ±0.03
MAX-MAX	1066.5 ±283.41	0.340 ±0.019	8.002	6.75 ±0.252	4.07 ±0.5	1.14 ±0.01
MAX-SUM	1003.9 ±191.74	0.621 ±0.028	8.002	6.60 ±0.249	4.05 ±0.52	1.12 ±0.08
MIN-CORR	969.65 ±243.9	0.688 ±0.034	8.002	6.57 ±0.239	4.08 ±0.56	1.12 ±0.01
MAX-CORR	576.63 ±124.80	1.288 ±0.059	7.99	7.42 ±0.147	3.63 ±0.61	1.24 ±0.03
MIN-CORR-VAR	900.00 ±160.11	0.552 ±0.018	8.002	6.63 ±0.23	4.00 ±0.51	1.13 ±0.01
MAX-CORR-VAR	915.83 ±235.25	0.716 ±0.022	7.99	7.59 ±0.18	3.59 ±0.48	1.27 ±0.03
HYPERBOLIC	376.97 ±79.73	787.9 ±46.65	7.90	6.82 ±0.25	4.11 ±0.51	1.16 ±0.013
MECA (benchmark)	2025.4 ±473.68	-	15.80	5.84 ±0.224	3.90 ±0.55	1

Table 4.2: Average ± standard deviation of the evaluation metrics for the different algorithms.

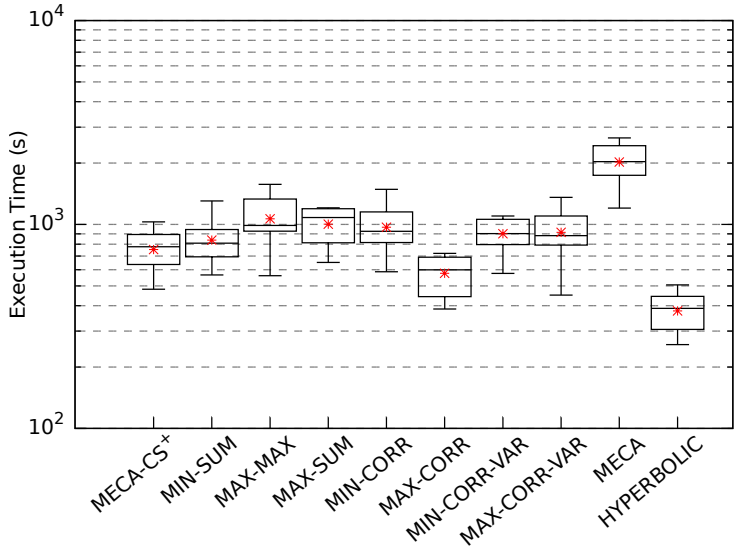
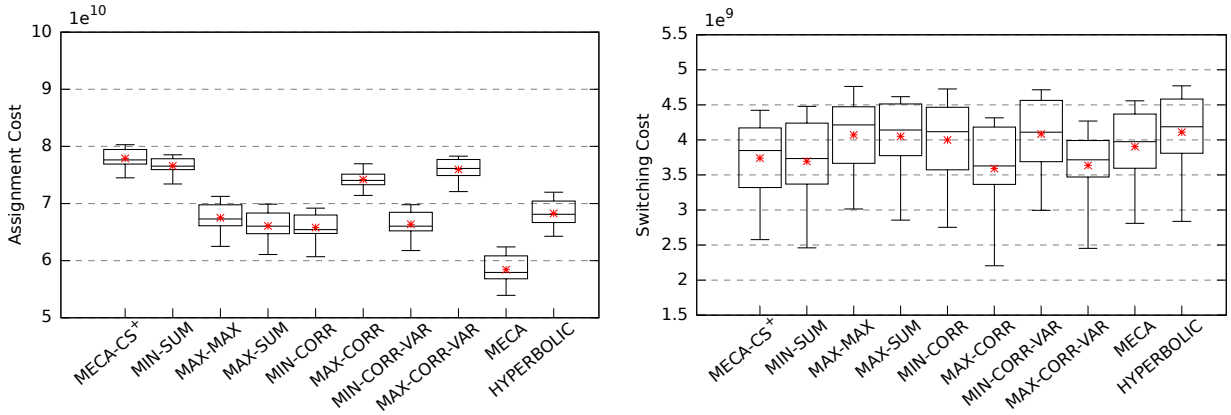


Figure 4.2: Orchestration execution time.

4.4.2.2 Maximum RAM usage

From Table 4.2, all the clustering-based algorithms are comparable in this respect; we record a slight difference between the HYPERBOLIC and all the other cases. We can remark that the benchmark algorithm is the most memory-consuming (almost the double than the others). We can also notice that the standard deviation is null (therefore omitted), i.e., changing the MEC hosts positions does not affect the maximum memory usage.



(a) Assignment costs.

(b) Swtiching costs.

Figure 4.3: Costs distribution

4.4.2.3 Assignment and switching costs

Figure 4.3 plots the distribution of both the assignment and the switching costs. Figure 4.3a shows that the benchmark algorithm (MECA), which is clustering-preprocessing free, has the lowest assignment cost. Indeed, as shown in Chapter 3, solving the orchestration problem for a set of clusters of APs forces the algorithms to propose the same orchestration plan for the AP belonging to the same MEC host, where some APs will not be assigned to the closest MEC host. Regarding the other algorithms, solutions grouping together APs with different demands through time have the lowest average assignment costs, i.e., MIN-CORR, MIN-CORR-VAR and MAX-SUM, whereas, the worst cases refer to the algorithms based on both MIN-SUM and MIN-MAX, which group together similar AP traffic demands. This shows that assigning APs with complementary demands profiles to the same MEC hosts reduces the assignment costs.

Figure 4.3b presents the distribution of switching costs. MAX-CORR-VAR and MAX-CORR, which group together APs with maximized correlation, have the smallest switching costs with an average of $3.59 \times e^9 \pm 0.48 \times e^9$ and $3.63 \times e^9 \pm 0.61 \times e^9$, respectively. On the other hand, HYPERBOLIC clustering has the highest switching costs values with an average of $4.11 \times e^9 \pm 0.51 \times e^9$. This shows that grouping APs with similar demand profiles allows having convenient assignment plans that last for longer periods of time compared to the other algorithms.

4.4.2.4 Convergence gap

Figure 4.4 shows the distribution of the convergence gap obtained by each of the aforementioned algorithms. By convergence gap we indicate the relative difference between the best solution (expressed by the total cost) produced by each of the clustering-based algorithms and the benchmark, within a given execution time limit.
The orchestration algorithms based on models that maximize the difference in demands (resp. minimize the correlation coefficient of APs of the same cluster) have the lowest total costs. We can also notice that these algorithms have the lowest assignment costs but not necessarily the lowest switching cost. This confirms that assigning APs with complementary demand profiles to the same cluster produces lower assignment costs.

4.4.2.5 BS-to-MEC distance

Figure 4.5 presents the distribution of access points to MEC facilities distances aggregated during a period of 1 week and for all the proposed MEC locations, for each of the algorithms. Benchmark gets the lowest distances, that is, the number of APs that are assigned to their closest MEC hosts is bigger compared to the other algorithms. As already explained, finding the closest MEC host to assign a single AP is easier than assigning a cluster of APs as all these APs should be have the same assignment plan.

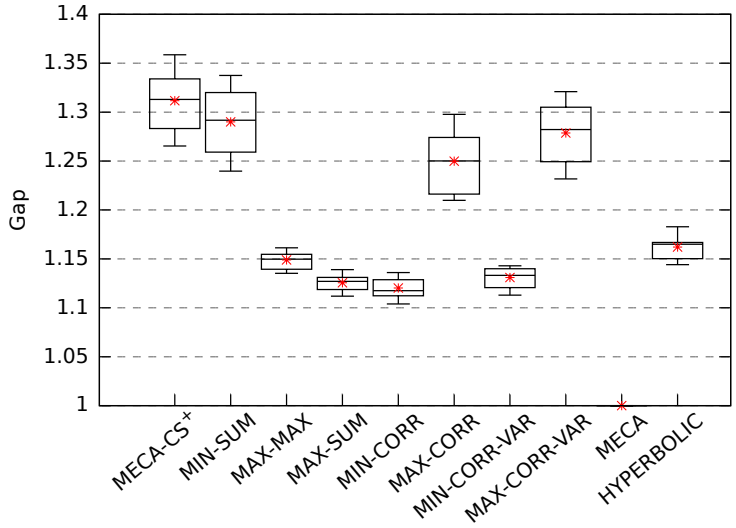


Figure 4.4: Gap ratio against Benchmark.

The other approaches are comparable, except HYPERBOLIC that gives slightly lower distances.

4.4.2.6 Switching operations

Figure 4.6 shows the number of switching operations on each MEC host during a period of 1 week. The benchmark gives the lowest number, followed by the algorithms that minimize the difference in traffic demands, i.e., MIN-MAX and MIN-SUM, while HYPERBOLIC yields the highest number of switching operations. In fact, this does not imply a lower switching cost, as can be seen in Figure 4.3b because it depends on the switched traffic demand of each of the base stations.

As a final remark on this part, it is worth stressing that assignment cost and execution time metrics are negatively correlated: the lower the execution times, the higher the assignment costs. For example,

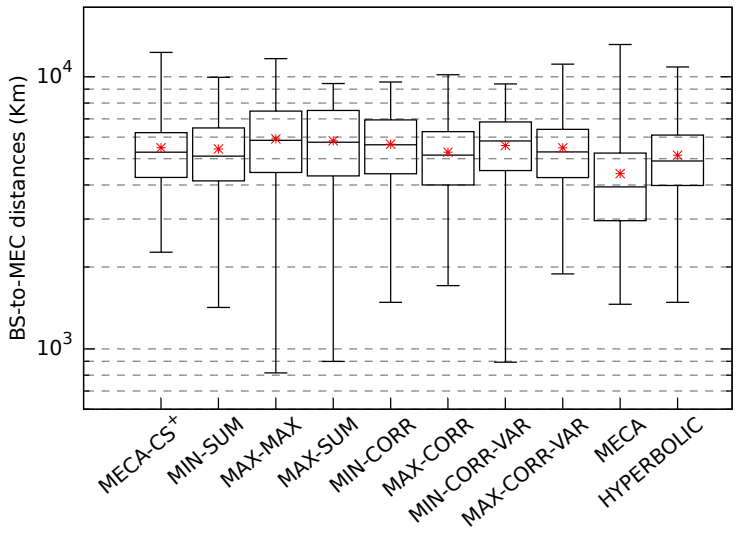


Figure 4.5: Distribution of the AP-to-MEC hosts distances.

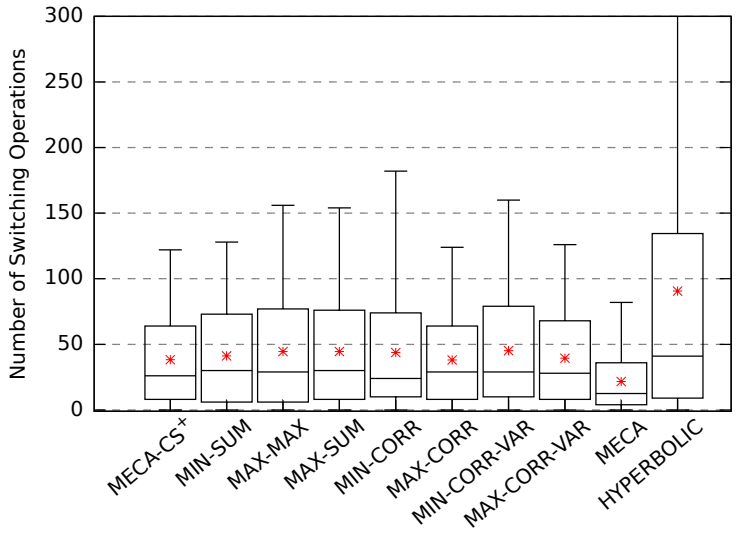


Figure 4.6: Distribution of the number of switching operations.

MAX-MAX requires the highest execution time when compared to the proposed solutions, but it yields lower assignment costs than MAX-CORR ($6.75e^{+10}$ vs $7.42e^{+10}$), where this latter represents the fastest approach. We can also notice that solutions generating the lowest switching costs, require less running time.

4.4.3 Robustness Analysis

In this part, we evaluate the robustness of the proposed solutions over time, i.e., the impact of applying the resulting orchestration plans on held-out (test) data. For that purpose, we evaluate the violation of MEC host capacity produced by each solution when applied to both the Lyon and Paris datasets.

For this purpose, we use the set of parameters initially defined in [34] in order to estimate the degree of controlling MEC hosts capacities by each algorithm. We assess the performance for different maximum occupancy rates achieved by MEC hosts, while changing the capacity sizes. We choose high utilization levels in the interval between 71% to 95.5%. The analyzed parameters are as follows:

Capacity Overload Average

$$\sum_{t\in T}\sum_{k\in K}\max\left\{\sum_{c\in C} d_c^t * x_{ck}^t - Q, 0\right\}/(Q\times \mid T \mid \times \mid K \mid) \tag{4.4}$$

This index (called SUM-SUM in [34]) gives the average of the demands exceeding the MEC host capacity over all the time periods.

Violation Rate

$$\mid \{(t,k) : \sum_{c\in C} d_c^t * x_{ck}^t - Q \geq 0, \forall t \in T, \forall k \in K\} \mid /(\mid T \mid \times \mid K \mid) \tag{4.5}$$

This index (called SUPPORT in [34]) computes the percentage of number of violations that occurred over all periods of times.

Excess Demand Average

$$\frac{\sum_{c\in C, k\in K : d_c^t * x_{ck}^t - Q \geq 0}\ \sum_{c\in C}(d_c^t * x_{ck}^t - Q)/Q}{\mid \{(t,k) : \sum_{c\in C} d_c^t * x_{ck}^t - Q \geq 0, \forall t \in T, \forall k \in K\} \mid} \tag{4.6}$$

This index (called SUM-SUM-SUPPORT in [34]) is used to show the relationship between the amount of excess demands and the total number of violations.

Note that our take-away on this aspect is that, in an operational carrier grade environment, when the actual assignment operation yields a capacity violation, the service is not interrupted but runs instead in a degraded mode, given the actual computing scheduler management of peak overloads by means of resource sharing policies. We represent the obtained results using the percentage gap with respect to the benchmark, computed as the ratio between the (i) difference between a given algorithm value and the benchmark value and (ii) the benchmark value.

For the sake of readability, we report only the results for the fastest algorithms from the previous analysis: MECA-CS$^+$, MIN-SUM, MAX-CORR, MIN-CORR-VAR and HYPERBOLIC.

4.4.3.1 Lyon dataset

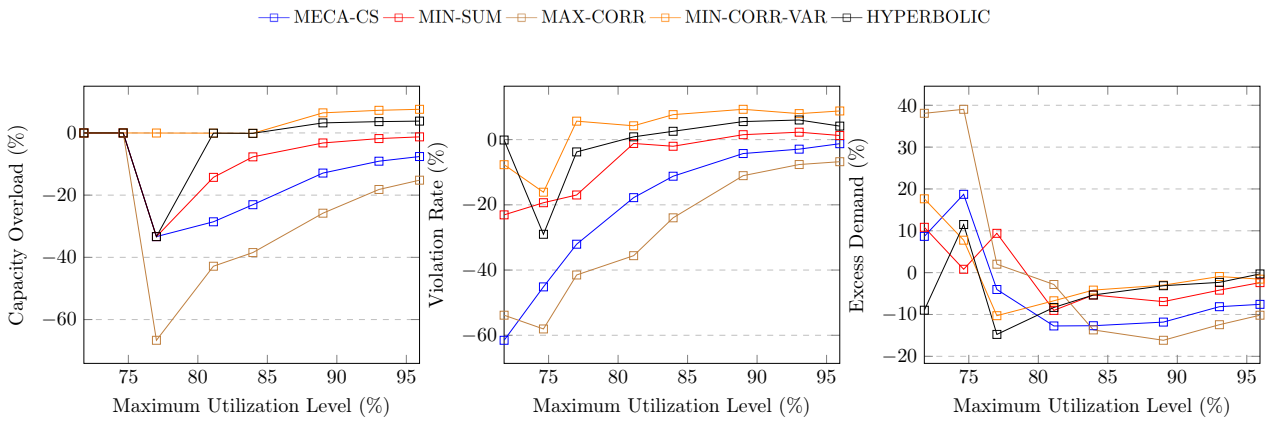
Figure 4.7 reports the robustness gaps with respect to the benchmark, as a function of the maximum MEC host utilization, for the five aforementioned approaches, for the Lyon dataset.

In terms of capacity overload (Figure 4.7a), we can observe that:

- When the maximum utilization is less than 75%, the gap is null: all the algorithms yield the same overload as the benchmark.
- For a maximum utilization level up to 84%, MIN-CORR-VAR yields the same overload as the benchmark and then increases it for higher utilization levels, whereas, all the other algorithms decrease the overload; the highest difference happens with an utilization level equal to 77% with a decrease of 68% for MAX-CORR algorithm and 34% for the two others (MECA-CS$^+$ and MIN-SUM).
- For a maximum utilization greater than 84%, cases merging complementary APs profiles (MIN-CORR-VAR and HYPERBOLIC) increase the capacity overload. This is reduced when using models that group together APs with similar demands (MECA-CS$^+$, MIN-SUM and MAX-CORR), i.e., these algorithms give an assignment with better robustness.
- MECA-CS$^+$, MIN-SUM and MAX-CORR lower the capacity overload when compared to the benchmark for each of the utilization levels greater than 75%. This shows that clustering AP demands in the preprocessing phase does not necessarily reduce the server capacity overload. In fact, the solution quality depends on the clustering criterion.

In terms of violation rate (Figure 4.7b), we can observe that:

- Except for HYPERBOLIC that yield the same number of violations for a maximum occupancy level equal to 71%, the other algorithms produce fewer violations when compared to the benchmark for a maximum utilization level less than 75%.
- MIN-CORR-VAR and HYPERBOLIC yield a higher number of capacity violations when the maximum occupancy ratio is greater than 77% and 81%, respectively, when compared to the benchmark. On the other hand, MIN-SUM produces less violations when the maximum level of utilization is less than 87%.
- Both MECA-CS$^+$ and MAX-CORR have a better violation robustness, i.e., they produce less violations through all the maximum levels of occupancy when compared to the benchmark and to the other algorithms. This can be justified by the fact that these two approaches propose assignment patterns for groups of APs with less variations on their total demands through time.



(a) Capacity Overload (average).

(b) Violation Rate

(c) Excess Demand (average)

Figure 4.7: Robustness results as a function of the maximum MEC host utilization - Lyon dataset.

Since it is easier to fit AP demands separately into their closest MEC hosts when training the model, the likelihood of reaching full MEC hosts capacity increases too when compared to clusters-to-MEC assignment. However, hosting high outlier demands generated in the test set data will be more difficult in the latter case above. On the other hand, the assignment of traffic demands using clustering-based algorithms during the training phase, tend to be more balanced as assignment operations should be

carried out for the demands of all APs belonging to the same cluster. This can justify the fact that clustering APs before their assignment to MEC hosts can help in reducing servers capacity violation when applied on held-out data.

In terms of capacity excess (Figure 4.7c), we can remark that:

- High values of the excess demands average refer to relatively low violation number compared to the amount of demand overload. Similarly, small values of the excess demands indicate that there is a relatively large number of capacity violations when compared to capacity overload. For a maximum capacity utilization between 71% and 75% all the models produce higher capacity excess on average when compared to the benchmark, except for HYPERBOLIC.

- For a maximum occupancy less than 76% and 79% when using MECA-CS$^+$, and MIN-SUM and MAX-CORRELATION, respectively, even though the capacity overload and the violation rate are both reduced compared to the benchmark, they produce higher average of excess of demands. This can be explained by the fact that these cases are generating relatively high excess demands, compared to the number of violations which makes the ratio bigger (in contrast to the benchmark, where the number of violations tend to be relatively low compared to capacity excess).

- For a level of occupancy greater than 79%, the excess is reduced when using clustering approaches, except for HYPERBOLIC that produces the same average of excess demands as the benchmark for the highest maximum utilization level. This can be explained by the fact that the clustering algorithms yield a proportionally higher number of violations in comparison to the demands excess (in contrary to the Benchmark that produces relatively high capacity overloads compared to the its number of violations).

- In fact, applying the resulting assignment on held-out demands shows that training the orchestration model with Lyon dataset using clusters built based on the similarity of their traffic demands produces assignment and switching plans that are more suitable for traffic fluctuations. Thus, MECA-CS$^+$, MIN-SUM and MAX-CORR outperform both the algorithms from the state-of-the-art where they reduce the capacity excess of demands for any maximum occupancy level.

4.4.3.2 Paris dataset

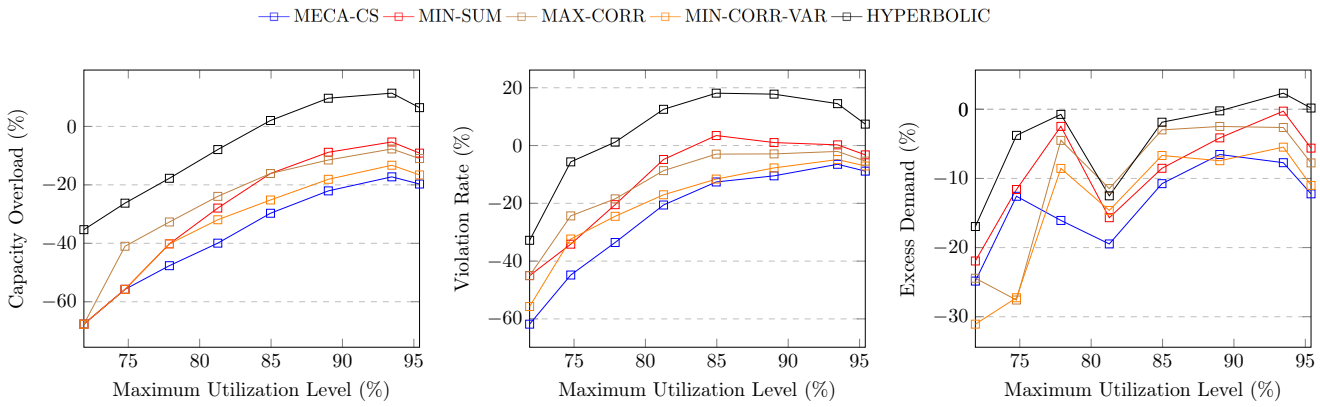
Figures 4.8 depict the robustness metrics results for the Paris dataset. In terms of capacity overload (Figure 4.8a) we have the following observations:

- Except for HYPERBOLIC, all other algorithms reduce the capacity overload of the MEC hosts for each given utilization level.
- The highest reduction occurs at a utilization of 71%, with a decrease of 68% for the observed approaches. On the other side, the lowest decrease is recorded when the utilization level is equal to 93.5% with a reduction of 18% for MECA-CS$^+$, 15% for MIN-CORR-VAR, 8% for MAX-CORR and 5% for MIN-SUM.
- When the maximum utilization is less than 85%, HYPERBOLIC lowers the overload demands compared to the benchmark and increases it for higher utilization levels, where it achieves the max growth of 11% when this latter is equal to 93%.

Looking at the violation rate (Figure 4.8b), we can assert that:

- MECA-CS$^+$, MAX-CORR and MIN-CORR-VAR reduce the number of violations for all the given utilization levels. As already explained, using clustering models at the preprocessing phase leads to finding the same assignment plan for each group of AP; fitting the total amount of their demands all together is then more difficult compared to single AP assignment problem. Thus, proposing solutions that ensure balanced availability on MEC hosts capacity is more likely to happen with cluster-based assignment. In addition, grouping access points based on their demands can help to assign groups of AP presenting less variance on their demands through time, which can justify the fact that some clustering-based algorithms outperform others.
- When the maximum utilization is less than 83%, MIN-SUM decreases the number of violations in comparison to the benchmark, and increases it for an utilization between 83% and 93%. As mentioned before, this latter produces less demands excess for such utilization levels. This can be justified by the fact that this approach generates high violations with low excess of demands.
- HYPERBOLIC decreases the number of violations by at most 30% when the servers maximum usage level is equal to 71%. On the other hand, for a capacity utilization level greater than 78%,

the number of violations raises when compared to the Benchmark. When the occupancy level is greater than 93%, we get a decrease in the number of violations for all the cases when compared to the benchmark.



(a) Capacity Overload (average). (b) Violation Rate (c) Excess Demand (average)

Figure 4.8: Robustness results as a function of the maximum MEC host utilization - Paris dataset.

Finally, the results in Figure 4.8c show that:

- Even though both the capacity overload and the violation number difference percentage values are monotonously increasing, there is a variability in the average of excess demands. This can be explained by the fact that this ratio is related to the amount of excess demands per each violation. Having a high violation number with low demand overload yields a larger average of excess demands and vice versa.
- All the clustering algorithms decrease the average of excess demands for all maximum utilization levels, except for HYPERBOLIC when this latter is greater than 87%.

All in all, the results provided in Figure 4.8 show the contribution of iterated pairwise clustering when integrated to the orchestration algorithm and trained using traces from a highly heterogeneous metropolitan area network as the Paris one. When the robustness against traffic fluctuations is higher, the number of violations and the excess of capacity are reduced even for the highest MEC infrastructure utilization levels. It is worth mentioning that, overall, MECA-CS$^+$ and MAX-CORR provide the most accurate results when applied to both Lyon and Paris datasets.

4.5 Conclusion

In this chapter, we proposed a collection of base station clustering models aiming at grouping base station APs in a robust way with respect to their assignment to edge computing facilities. By leveraging on state-of-the-art orchestration algorithms for the assignment problem, we have evaluated the performance of the proposed approaches using real-world traffic demands, while comparing them to other two state-of-the-art approaches. Through extensive simulations and evaluation in terms of different performance metrics, we show under which conditions the algorithms we propose reveal to be the most efficient ones. We further compared the four fastest clustering algorithms with two state-of-the-art algorithms in terms of robustness against traffic fluctuations, and using two different city datasets. Among many important observations showing the general superiority of our approaches with respect to the state-of-the-art ones, a promising finding is that the robustness of the algorithms is higher with larger traffic source diversity.

Chapters 3 and 4 focused on the resource orchestration of MEC infrastructures, more precisely on the problem of assigning BSs to MEC hosts. Another interesting perspective related to MEC orchestration is to consider resource orchestration at the application level in MEC environments. To this end, we present in the following chapter a placement model for distributed edge-AI applications in a federated learning environment.

Chapter 5

Optimal Placement of MEC Applications for In-network Federated Learning

Contents

5.1 Introduction

The integration of artificial intelligence modules in network components is been happening since less than a decade. Modern network nodes nowadays integrate Neural Processing Units (NPUs), ranging from mobile devices to core backbone equipment. The community often speculated that in the beginning of this trend, vendors did not know for which applications these units would be useful, but gambled on their future usefulness.

Network automation is expected to be one of the applications that could leverage on distributed AI modules for both learning and inference tasks. New network functions related to analytics tasks have already appeared in telecommunication standards, as for instance the NetWork Data Analytics Function has made surface in 3GPP 5G system since Release 16 [74] as a function tailored to the analysis of monitoring data from the 5G core network functions. Indeed, the 5G core network function cluster is being increasingly integrated in core networks, with a much higher level of geographical distribution, mostly led by the need to offer 1 ms access latency performance to 5G services.

Such performance targets are therefore pushing for distribution of network functions and related monitoring, learning and inference tasks. For this re-architecturing, 5G and beyond-5G solutions are leveraging on multi-access edge computing (MEC) technologies, with so-called traffic local break-out gateway to steer some traffic requiring such performance to edge application servers co-located with distributed 5G core functions [75]. On the other hand, the MEC architecture host servers that, besides serving end-application needs, could also serve to deploy computing functions for a set of infrastructure needs. The AI@EDGE H2020 European project [76] delved into the definition of AI Functions meant for distributed learning and inference functions, serving both application and infrastructure (e.g., NWDAF) computing needs, and meeting stringent 5G and beyond-5G performance requirements in terms of latency, impairment detection and feature prediction.

Few distributed learning frameworks have made their way into commercial computing systems. Among them, Federated Learning (FL) introduces a form of hierarchical learning where edge nodes perform learning based on data which stays in the locality of the edge nodes, and send the result for their local learning to a server, which aggregates the learning parameters of multiple edge nodes and then updates the edge nodes with its global view parameters. Besides being already used for a number of mobile device usages by companies as Google and Meta [77], FL is also being considered

for in-network AI functions as the NWDAF [46, 47, 74]

In this chapter, we address the problem of placing AIFs running federated learning against connect-compute network infrastructure monitoring data, for environments where the introduction of edge computing comes with a heterogeneous and large set of computing and networking elements, requiring low latency performance. Among the multiple distributed learning techniques proposed in the literature, federated learning [45] reveals as a good compromise between the need to distribute the learning and guarantee a centralized view in support of efficient inference, and receives large industry support and integration, including in 3GPP standards [44].

We rely on a federated learning anomaly detection AIF system proposed in [12], adapted for the 5G infrastructure which makes use of a centralized server AIF and a variable number of edge AIFs while load-balancing monitoring data among the edge client AIFs. In this domain, our contributions are as follows[1]:

- We propose a MILP formulation for the placement of AIFs using a federated learning setting. The proposed model takes into consideration (i) the FL server location, (ii) the latency budget covering learning and communication delay components, and (iii) the respect of a target learning time;
- We assess the impact of using hardware accelerators on a subset of edge nodes in order to reduce the training time, and how the deriving latency unbalance can be compensated in the placement outcome;
- We compare the proposed approach to two different variants of the AIF placement model in terms of achievable learning time and number of deployed AIFs.

We present the reference distributed anomaly detection AIF system in the following.

AI-enabled end-to-end applications

We introduce the novel concept of Artificial Intelligence Functions (AIFs) to refer to the AI-enabled end-to-end applications sub-components that can be deployed across edge-enabled B5G and 6G

[1]Part of this work was published in the 2022 18th International Conference on Network and Service Management (CNSM) [78]

networks. In such a distributed learning setting, multiple AIFs need to be distributed close to, or at the place where monitoring data is generated, to run distributed continuous learning in support of low latency runtime inference.

The functional architecture of an AIF is given in Figure 5.1. To support its operation, we identify five interfaces:

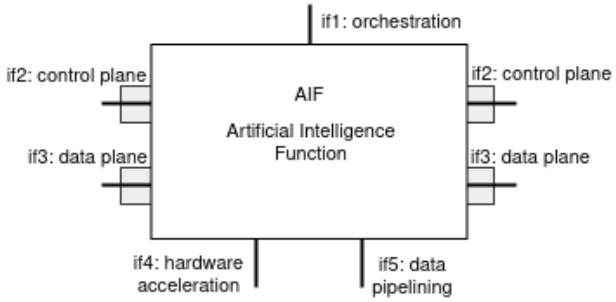


Figure 5.1: AIF reference representation.

- interface (if)1: used by the orchestration platform for the communication with the AIF, including its configuration (e.g. for dynamic update of federated learning hyper-parameters) and the retrieval of inference results (e.g., inference running at the server AIF and/or edge AIFs);
- if2: used for AI model parameter exchange among AIFs (AIF control plane interface), e.g., the communication between edge AIFs and server AIF in federated learning;
- if3: used for data exchange among different AIFs (AIF data-plane interface) - which may be used for generic distributed learning, in the case of an AIF forwarding graph;
- if4: hardware acceleration interface with components as GPU and Smart-NICs, for training and inference tasks;
- if5: for data collection and streaming, to interface with a data-pipe-lining system.

A distributed AIF system making use of FL is depicted in Figure 5.2: via if2, edge AIFs send local training results and obtain global training parameters back from the FL server AIF. if3 is unused in FL AIFs. if5 makes use of a data pipelining system for getting data for AIFs. The usage of HWAs as NPU via if4 is meant to accelerate training and inference tasks, where inference could possibly be

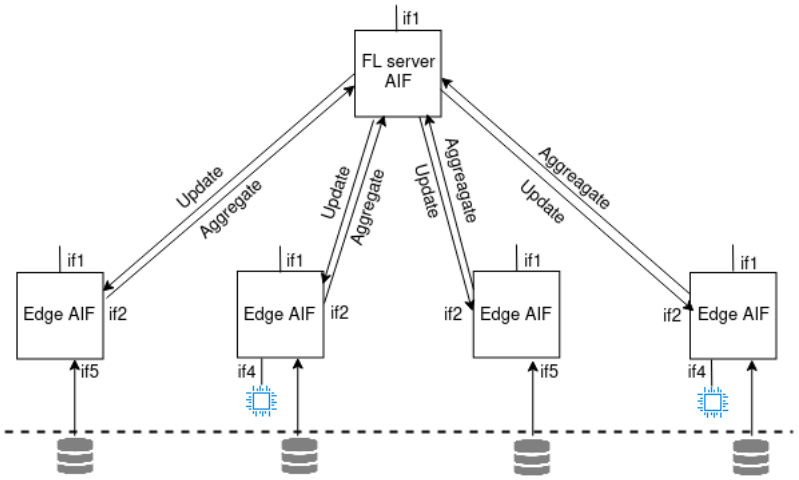


Figure 5.2: FL-based anomaly detection AIF systems.

taking place at the edge AIF level besides the server one. An example of in-network FL application is anomaly detection for network automation loops. In this application, outlined in [12], distributed AIFs make use of Long-Short Term Memory (LSTM) autoencoders against group of metrics related to network, storage, operating system features, to spot anomalous behavior. The goal in such application is to support automatic reconfiguration of the infrastructure stack (e.g., rescaling, load-balancing, rerouting) based on the detected anomaly fingerprint.

5.2 Problem Statement

We define the AIF placement as the problem of finding the optimal number of AIFs and their location on a given network graph, taking into account the inter-AIF communication patterns, target learning loop time performance and the presence of hardware acceleration.

More precisely, with respect to the reference AIF model (Figure 5.1) in this work we consider that:

1. communication with the orchestration platform (if1) is possible at any node where AIF can be placed;
2. AIFs interact using federated learning, hence using the if2 interface for exchanging AI model parameters;
3. if3 is not used, because in federated learning raw data is not exchanged among nodes;
4. if4 is enabled only at a subset of the candidate AIF location for hardware acceleration;

5. data pipelining delay (if5) is negligible;

6. the propagation delays over the link between candidate edge AIF locations and server AIF are not negligible with respect to the learning time;

7. the servers hosting AIFs offer them the same computing capacity.

The optimization goal is the placement of the FL server and clients that are both deployed as AIFs while respecting the imposed target time.

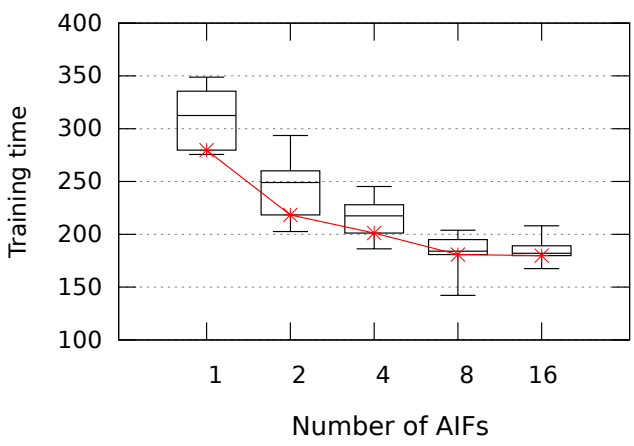
5.2.1 Empirical federated learning time distributions

In order to build a purpose-built model of the learning time as a function of the number of federated learning nodes that are used, we run the FL-based anomaly detection framework proposed in [12]. To do so, we run a set of AIFs on a Kubernetes infrastructure [79] where the placement is done automatically using kubernetes scheduler. In fact, each AIF is an implementation of a LSTM model. The model is composed of a set of autoencoders that allow to detect anomalies at different system levels, i.e., physical level, virtual level and access level, using different groups of metrics, e.g. CPU, memory and radio metrics.

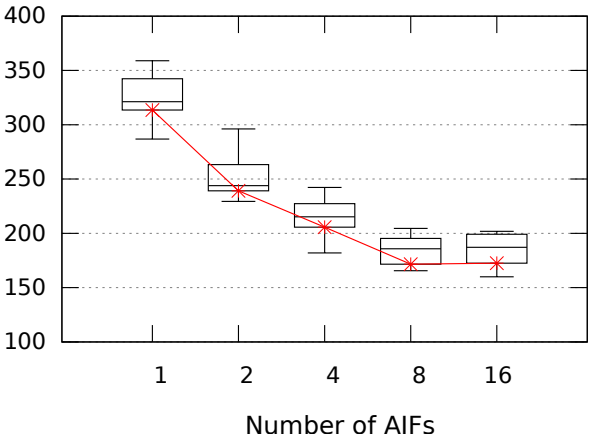
We use the 5G3E dataset from [80], providing few dozens of feature time-series for each resource group, where groups are CPU, RAM, network link and storage resources, and RAN and UE nodes. We train the ML model using data batches of 800 samples each, corresponding to 2 minutes of collected data for each data batch: this is the assumed retraining time of the system, which could vary in general depending on the sampling rate. The batch size is set to the data size, hence considering all the samples. The number of epochs is 10 and the model is trained for one round. The rest of the FL-based anomaly detection AIF hyper-parameters are the same as in [12].

In Figure 5.3, we present the corresponding training time distribution as a function of the number of active AIFs for four different resource groups features related to CPU, Memory, Network and File system (collected at the container level).

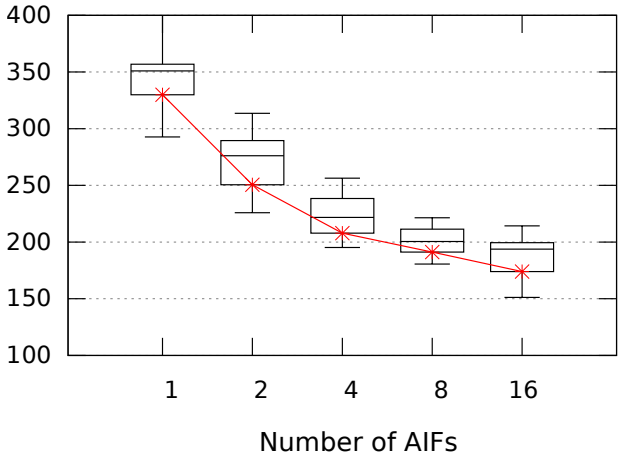
We can remark that the training time decreases with the increase of the number of AIFs up to a certain threshold value. A certain variance exists, due to CPU scheduling and storage system systemic variations at the operating system level. We employ in the model the piece-wise linear function



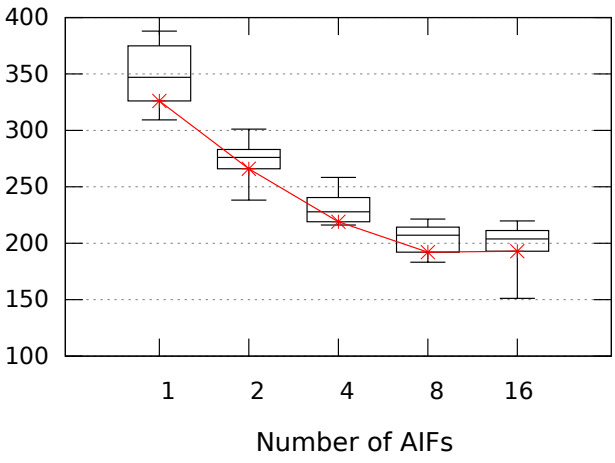
(a) CPU resource



(b) Memory resource



(c) File system resource



(d) Network resource

Figure 5.3: Training time (in ms) vs number of AIFs. $R = 1$, $E = 10$.

obtained using the first quartile values, indicated in red in the figures. In the following, we show how we rely on the obtained results to build the AIF placement model.

Sets and parameters	
$\|N\| = n$	set (and number) of physical servers
$\mathrm{p_k}$	max. distributed training time with k deployed AIFs
$\tilde{\mathrm{p}}_\mathrm{k}$	reduction in training time passing from $k-1$ to k AIFs
R	number of rounds of FL algorithm; i.e. number of times the AIFs exchange the model parameters with the FL server before the inference phase
T	target distributed learning time
$\mathrm{d_i}$	communication latency between the FL server and node i
D	maximum accepted latency between the FL server and an AIF
Variables	
x_i	binary variable with value 1 if an AIF is activated on node i
$t \geq 0$	distributed training time of an AIF

Table 5.1: Mathematical models notations.

5.2.2 Problem formulation

We consider a set N of physical servers such that an AIF can be hosted by any server. The communication latency between edge AIFs and the FL server AIF depends on the placement decision. Whereas, the distributed training time t is a decreasing function of the number of AIFs, and does not depend on the AIF location, as we suppose the same computing capacity is delivered to AIFs independently of the location. The main goal is to minimize the number of active AIFs used for training, while guarantying that the overall training loop time (including distributed edge AIF training time and edge-server AIF delay) is at most T, or equivalently, that the time of a single FL round is not longer than $\frac{T}{R}$.

In fact, the time of a single round depends on the distributed training time t and the transmission latency. We denote by d_i the transmission latency of an AIF hosted by the server $i \in N$. Additionally, p_k represents the distributed processing time when k AIFs are installed. This parameter is defined using a linear approximation on the values given by Figure 5.3. Consequently, for each AIF $i \in N$, the overall learning time of a single round when k AIFs are active is $p_k + max_{i \in N} d_i$. Note that at least two AIFs are enabled to ensure the distribution of the training task.

5.2.2.1 Greedy solution

If there exist a feasible solution, an optimal solution can be constructed as follows. At each iteration, we check if we attain the target time, if not, an AIF is added. The algorithm stops when the target time is attained or all the AIFs are used (eventually signaling that the problem is unfeasible). Even if this version of the AIF placement problem can be solved by a greedy algorithm, we present a MILP formulation. This will allow us to introduce more advanced variants of the problem.

5.3 Artificial Intelligence Function Placement

In the following, we present the mathematical formulation of the AIF placement problem then we propose two variants of the problem.

5.3.1 Mathematical model

We introduce an auxiliary parameter $\tilde{p}_k$ representing the gain in processing time passing from $k-1$ to k AIFs: $\tilde{p}_k = p_k - p_{k-1}$. If, by convention, we set $\tilde{p}_1 = p_1$, the processing time when k AIFs are active can be calculated as follows: $p_k = \sum_{i=1}^{k} \tilde{p}_i$. In Table 5.1, we recap the necessary notations used to introduce the mathematical formulation.

Following the greedy algorithm idea, the mathematical model defined by equations (5.1) to (5.5) assumes that the set N is ordered by increasing latency.

$$\min \sum_{i \in N} x_i \tag{5.1}$$

$$x_i \geq x_{i+1} \quad \forall i \in 1..n-1 \tag{5.2}$$

$$x_2 = 1 \tag{5.3}$$

$$t = \sum_{i=1}^{n} \tilde{p}_i x_i \tag{5.4}$$

$$t + d_i \leq \frac{\mathrm{T}}{R} + (1 - x_i)(p_1 + D) \quad \forall i \in N \tag{5.5}$$

$$t \geq 0$$

$$x_i \in \{0, 1\} \quad \forall i \in N$$

The objective in (5.1) minimizes the number of activated AIFs. Constraints (5.2) impose that the activation of the AIFs is done in increasing latency order, whereas constraint (5.3) imposes that at least two AIFs are installed to ensure the training task in a federated manner. Constraint (5.4) allows to calculate the distributed learning time (t can be removed by substitution, but we kept it for the sake of clarity). Constraints (5.5) guarantee that for each activated AIF the time for a single round is below $\frac{T}{R}$. If the AIF is not activated, i.e. $x_i = 0$, the constraint is always valid.

We now extend the model to consider the possibility of using a hardware accelerator to reduce the distributed learning time of an AIF. We assume that only a set of physical nodes are provided with hardware accelerator. We introduce three parameters: an indicator parameter h_i, its value is 1 if a hardware accelerator is available on node i, and 0 otherwise; the maximum number of hardware accelerators H, and the acceleration factor $0 < \alpha < 1$. If a hardware accelerator is installed, the processing time p reduces to αp. In this case, a full ordering cannot be obtained independently from the number of installed AIFs. Indeed, the processing time is influenced by the possibility of installing hardware accelerators and their available number (note a polynomial algorithm to solve this problem can be constructed).

To extend the baseline formulation, it is necessary to decouple the counting variable from the installation variable. Indeed, we cannot determine an ordering independent from the number of selected AIFs. From now on, no ordering is imposed on N. We introduce a binary variable z_k, with $k = 1..n$ that counts the number of activated AIFs. Note that there is no correspondence with the indexing of the two variables z and x. Furthermore, we introduce two additional sets of variables: a binary variable w_i that is equal to 1 if the hardware accelerator installed on node i is activated and used, 0 otherwise; a non-negative continuous variable δt, that represents the gain in processing time if the hardware accelerator is used, i.e, $\delta t = (1 - \alpha)t$. The formulation is as follows:

$$\min \sum_{i \in N} x_i \tag{5.6}$$

$$z_k \geq z_{k+1} \quad \forall k = 1..n-1 \tag{5.7}$$

$$z_2 = 1 \tag{5.8}$$

$$t = \sum_{k=1}^{n} \tilde{p}_k z_k \tag{5.9}$$

$$\sum_{k=1}^{n} z_k = \sum_{i \in N} x_i \tag{5.10}$$

$$t - \delta t_i + d_i \leq \frac{\mathrm{T}}{R} + (1 - x_i)(p_1 + D) \quad \forall i \in N \tag{5.11}$$

$$\delta t_i \leq (1 - \alpha)t \quad \forall i \in N \tag{5.12}$$

$$\delta t_i \geq (1 - \alpha)t - (1 - w_i)(1 - \alpha)p_1 \quad \forall i \in N \tag{5.13}$$

$$\delta t_i \leq w_i(1 - \alpha)p_1 \quad \forall i \in N \tag{5.14}$$

$$w_i \leq h_i \qquad \forall i \in N \tag{5.15}$$

$$\sum_{i \in N} w_i \leq H \tag{5.16}$$

$$t \geq 0 \tag{5.17}$$

$$\delta t_i \geq 0 \quad \forall i \in N \tag{5.18}$$

$$x_i, w_i \in \{0, 1\} \quad \forall i \in N \tag{5.19}$$

$$z_k \in \{0, 1\} \quad \forall k = 1..n \tag{5.20}$$

The objective function in (5.6) is unchanged. Constraints (5.8)-(5.9) use the counting variable z_i and substitute constraints (5.3)-(5.4). Constraint (5.10) allows to count the number of active AIFs, linking z and x variables. Constraints (5.11) are the updated version of constraints (5.5), where we take into account the time reduction due to the hardware accelerator, if installed and activated. The reduction is calculated using the auxiliary variable δt_i. The value of δt_i must be $(1 - \alpha)t$ if the hardware accelerator is installed, 0 otherwise. Constraints (5.12)-(5.14) allow to represent it in a linear form. Finally, constraints (5.15) limit the installation of hardware accelerators to the available nodes, and constraint (5.16) limits the number of installed hardware accelerators.

The next extension offers the possibility to choose the position of the FL server, which has an impact on the latency of communication with the active AIFs. The possibility of using hardware

accelerators (in a limited number), opens the following question: could it be cost effective to increase the latency of some AIFs to reduce the latency of others? In fact, the answer depends on several factors, the accelerator factor and the location of possible hardware accelerators, their maximal number and the impact of the location of the FL server on the transmission latency of the different installed AIFs.

$$\min \sum_{i \in N} x_i \tag{5.21}$$

equations $(5.8) - (5.10)$

$$t - \delta t_i + \sum_{j \in N} d_{ij} \xi_{ij} \leq \frac{\mathrm{T}}{R} + (1 - x_i)(p_1 + D) \quad \forall i \in N \tag{5.22}$$

equations $(5.12) - (5.16)$

$$\sum_{j \in N} y_j = 1 \tag{5.23}$$

$$y_i + x_i \leq 1 \quad \forall i \in N \tag{5.24}$$

$$\xi_{ij} \leq y_j \quad \forall i \in N, j \in N \tag{5.25}$$

$$\sum_{j \in N} \xi_{ij} = x_i \quad \forall i \in N \tag{5.26}$$

var. domain $(5.17) - (5.20)$

$$y_i \in \{0, 1\} \quad \forall i \in N$$

$$\xi_{ij} \in \{0, 1\} \quad \forall i \in N, j \in N$$

We extend the previous formulation adding two sets of variables: a binary variable $y_j, j \in N$, that takes value 1 if the FL server is installed on node j and 0 otherwise, binary variable $\xi_{ij}, i \in N, j \in N$, that takes value 1 if an AIF is installed on node i and the FL server is installed on node j, and 0 otherwise. The latency parameter d_i is then substituted by its extended version d_{ij} that depends on the nodes i and j on which the AIF and the FL server are respectively installed. In equations from (5.21) to (5.26) we report the extended formulation, highlighting the changes with respect to the previous one.

Constraints (5.22) take into account the delay due to the location of the FL server. In constraints (5.23) we guarantee that only one FL server can be installed on a given physical node. In fact, the node hosting the FL server cannot host an AIF (see constraint (5.24)). Constraints (5.25)

ensures that for each node i, the only ξ that can be activated is the one corresponding to the node of the installed server. Finally, coherence between ξ and x variables is enforced by constraints (5.26)[2].

5.3.2 Variants of the AIF placement problem

In the following, we present two different variants for the above placement problem.

5.3.2.1 Minimal FL update arrival time variance

We propose a variant of the placement model in which we minimize the difference between the highest and the lowest learning loop times, where a learning loop encompasses edge AIF training and the transmission of the new learning metrics to the FL server. The main objective of this model is to reduce the straggler effects [81], where the aggregation after each round of training depends on the slowest AIF: data coming too late at the FL server because of too long propagation delay, or too long edge AIF training time, or both combined, is not counted at a given round, hence decreasing the training quality.

We add two variables $maxT$ and $minT$, representing respectively the highest and the lowest learning time produced during the training taking into account the communication delay with the FL server. This relationship is represented by the the following constraints:

$$maxT \geq t - \delta t_i + \sum_{j \in N} d_{ij} \xi_{ij} \quad \forall i \in N \tag{5.27}$$

$$minT \leq t - \delta t_i + \sum_{j \in N} d_{ij} \xi_{ij} \quad \forall i \in N \tag{5.28}$$

Then, we introduce a new term in the objective function which represents the difference between the maximum and the minimum training time. The new objective function is presented by equation (5.29).

$$\min A \cdot \sum_{i \in N} x_i + B \cdot (maxT - minT) \tag{5.29}$$

[2]using these constraints, variables x can be removed

5.3.2.2 Minimal target learning time violation

In this variant, we accept solutions even when the target learning time is violated. We introduce a continuous variable U that represents the fraction of the achieved target learning time for the proposed placement solution.

The new objective in (5.30) aims to reduce the achieved overall learning time by minimizing the value of U in order to respect the target learning time.

$$\min \quad U \tag{5.30}$$

Also, equation (5.22) is replaced by (5.31). This constraint allows us to have feasible placement solutions even if the target learning time is exceeded.

$$t - \delta t_i + \sum_{j \in N} d_{ij} \xi_{ij} \leq \frac{U \cdot \mathrm{T}}{R} + (1 - x_i)(p_1 + D) \tag{5.31}$$

5.4 Experimental results

In the following, we detail the experimental setting, then we provide an evaluation of our placement model along with a comparison of the different model variants.

5.4.1 Simulation setting

We use the Mandala topology from [82]: it consists of connecting access nodes through three tiers, i.e., aggregation, core and application (i.e., egress) nodes. For instance, one may consider the edge servers as MEC hosts in a MEC system in a Metropolitan Area Network topology or a near edge AIF deployment (see Figure 5.4). The total number of nodes is equal to 26 including 16 edge nodes.

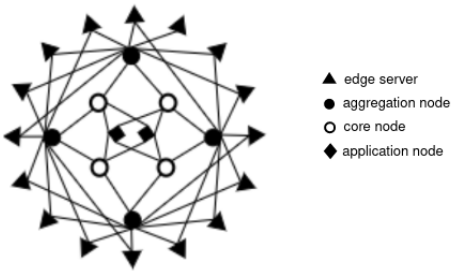


Figure 5.4: Mandala topology [82].

We solve the AIF placement problem using the following algorithms

- Baseline: greedy algorithm presented in Section 5.2.2;
- AIF-P: placement MILP algorithm (equations from (5.21) to (5.26)), including hw accelerators;
- AIF-P-nohw: same as AIF-P but without using hardware accelerators;
- AIF-P-var: using the first variant of the placement model presented in Section 5.3.2.1. After several preliminary tests with different values of A and B, we opted for $A = 1$ and $B = 10^3$. This in fact makes AIF-P-var behave differently from AIF-P;
- AIF-P-U: using the second variant of the placement model presented in Section 5.3.2.2.

We propose to analyze the impact of the following parameters on the behavior of the five aforementioned algorithms:

- Number of rounds: we test different rounds of FL training, i.e., 1, 5, 10, 15 and 20, before the inference step. In fact, increasing the number of rounds helps increase the quality of the learning.
- Target learning time: we evaluate the proposed algorithm for different target times, i.e., 2 s, 1.6 s and 1.2 s. This parameter specify the time during which we train the model, before its exploitation for inference.

In order to evaluate the proposed solutions, we generate 15 different configurations for both the placement and the number of hardware accelerators, such as the number of available accelerators is randomly generated within the interval of [10%,60%] of the total number of nodes, in order to analyze the impact of their availability on the proposed solution.
The latency on the links is randomly chosen such that the highest round-trip time for the shortest path between the most distant nodes is equal to 7% of the lowest training time, with a small part of communication delay with respect to the training time. The acceleration factor α is set to 0.5, so that the training time reduction is not too large nor too small.

5.4.2 Results analysis

In the following, we present the simulation results.

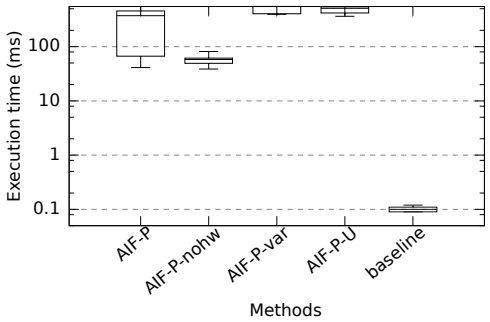


Figure 5.5: Execution time for the different algorithms.

5.4.2.1 Execution time

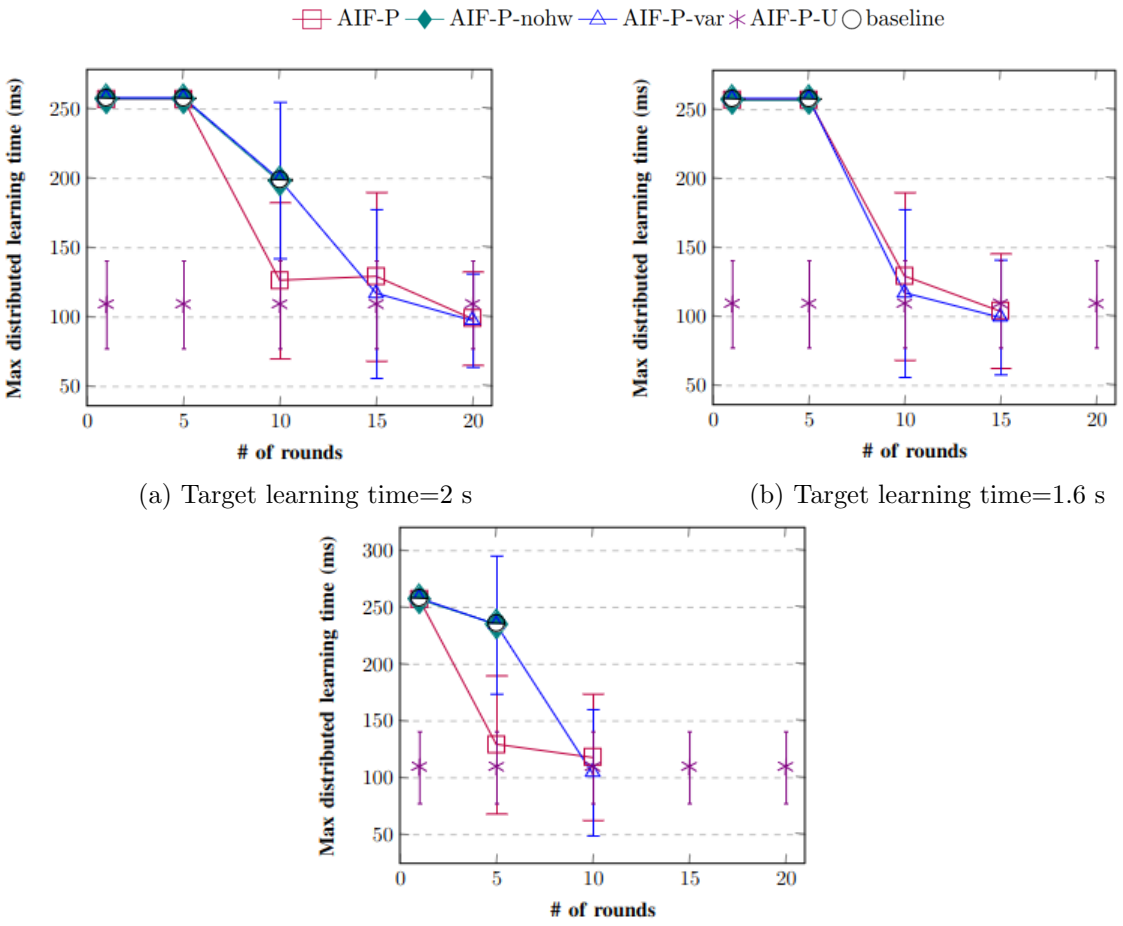
In Figure 5.5, we present the distribution of the execution time needed by each of the aforementioned algorithms for all the modeling settings.

As expected, the baseline greedy algorithm is the fastest one, with an execution time below 1 ms, followed by AIF-P-nohw; this can be explained by the fact that AIF-P-nohw does not decide on hardware accelerators in addition to the function placement. Even if the two algorithms produce lower execution times when compared to the other approaches, unfeasible solutions are frequently generated as soon as the time constraints become a bit tighter. We can also notice that AIF-P-U and AIF-P-var undergo a similar execution time, and are the slowest ones. AIF-P is in between the latest three approaches.

5.4.2.2 Edge AIF learning time

In Figure 5.6, we present the average of the maximum edge AIF distributed learning time (maximum among all the feasible solutions) for the different 15 hardware accelerator configurations, for different target times (5.6a, 5.6b and 5.6c) and using different numbers of rounds. The error bars represent the standard deviation. We can notice that:

- The distributed learning time decreases with the increase of the number of rounds for all target times. as already discussed in Section 5.2.1; this can be explained by the fact that increasing the number of rounds requires higher number of active AIFs at each round since the increase of this latter decreases the training time.
- In Figure 5.6a, we can notice that with 15 rounds, and unlike all the other cases, AIF-P suffers



(a) Target learning time=2 s

(b) Target learning time=1.6 s

(c) Target learning time=1.2 s

Figure 5.6: Max learning time vs number of rounds (lines for the baseline and AIF-P-U are omitted for the sake of clarity).

a slight increase in the learning time when compared to 10 rounds. This likely happens because we show the maximum distributed learning time over all the feasible solutions, hence unfeasible solutions are not taken into account: instances that yield solutions with high learning times using 10 rounds are not accounted in the case of 15 rounds since the algorithm gives unfeasible solutions for this specific case.

- AIF-P-U produces the same average learning times for all the cases, regardless of the number of rounds and the target learning time. In fact, it deploys the same solution with all the nodes having accelerators, for all the different parameters even if it is not necessary in some cases (i.e., deployment of more AIFs than needed). However, if the target learning time is violated, AIF-P-U still leads to a placement solution. This approach yields the lowest (equivalent, in some cases) distributed training time when compared to all the other algorithms, for all the target times.

- Table 5.2 shows the violation rates U for all the proposed settings. In fact, with less strict time constraints $U < 1$, while U increases with the increase of the number of rounds and the target learning time. Differently than the other approaches, AIF-P-U yields values higher than 1 instead of unfeasible solutions when the target learning time is violated.

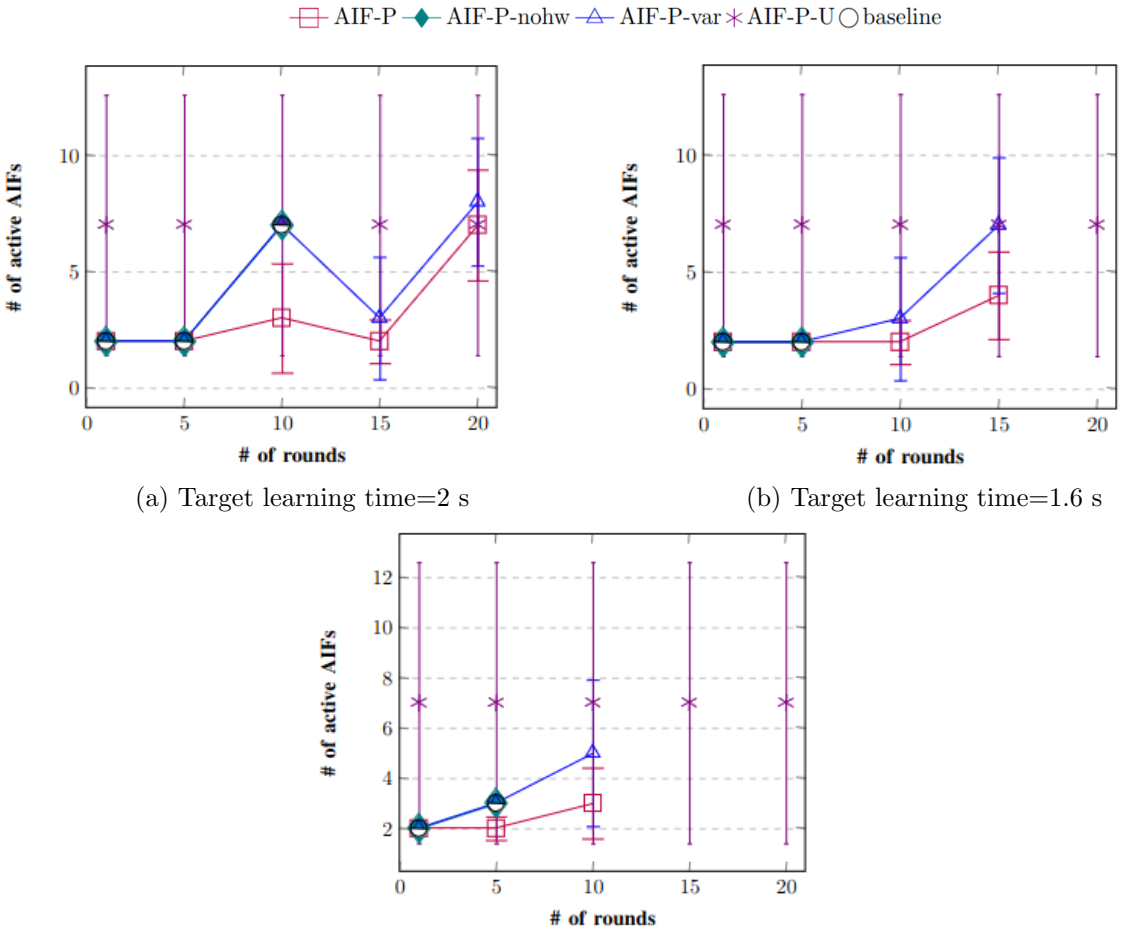
- We can also notice that AIF-P and AIF-P-var have similar behaviors: they can produce a placement solution for all the proposed numbers of rounds for a target learning time equal to 2 s. On the other side, both algorithms are not able to find any feasible solution for some stringent time constraint cases; these unfeasible solutions correspond to a number of rounds higher than 15 (respectively 10) for a target time equal to 1.6 s (respectively 1.2 s).

- For all the proposed target times, AIF-P produces a lower distributed learning time than AIF-P-var when $R = 10$ for $T = 2s$ and T = 1.6 s (R = 5 for T = 1.2 s respectively). However, this is not the case anymore when R increases. Differently than AIF-P-var, in order to respect the learning time threshold, AIF-P tends to use hardware accelerators instead of increasing the number of deployed AIFs.

- Both the baseline and the AIF-P-nohw show less performance when compared to the other methods: any feasible solution is found for a number of rounds that surpasses 10 and 5 when the target time is equal to 2 s (5.6a), and 1.6 s and 1.2 s respectively (5.6b and 5.6c).

- For the baseline, a very small variation in the solutions was recorded when the T = 1.2 s using 1 and 5 rounds.

These results confirm the usefulness of hardware accelerators during the training phase. In fact, the reduction in the distributed learning time in order to achieve 20 rounds (respectively, 15 and 10 rounds) when T = 2 s (T = 1.6 s and T = 1.2 s, respectively) is between 50% and 59% for all the three approaches using hardware acceleration.



(a) Target learning time=2 s

(b) Target learning time=1.6 s

(c) Target learning time=1.2 s

Figure 5.7: Number of active AIFs vs number of rounds (lines for the baseline and AIF-P-U are omitted for the sake of clarity).

	R=1	R=5	R=10	R=15	R=20
T=2 s	0.7	0.32	0.65	0.97	1.29
T=1.6 s	0.08	0.40	0.81	1.21	1.61
T=1.2 s	0.11	0.54	1.08	1.61	2.15

Table 5.2: Achieved violation rates U.

5.4.2.3 Number of active AIFs

In Figure 5.7, we present the average number of active AIFs using the different algorithms, while varying both the number of rounds and the target learning time. The length of the error bars represents the variation in the number of AIFs produced by the different 15 simulations for the same case, i.e, same number of rounds and the same target learning time.

- As expected, the plots show an inverse relationship between the distributed learning time (Figure 5.6) and the number of deployed AIFs.
- We can notice that, beside AIF-P-U which has a constant behavior regardless of the changing in time constraint, all the other algorithms show an increase in the number of AIFs with the increase of of time constraints, except for the case of 15 rounds with a target time equal to 2 s using AIF-P and AIF-P-var shown in Figure 5.7a. As already explained, we present the average of the obtained values from all feasible solutions, and where some configurations with 10 rounds require a high number of active AIFs to respect the target learning time. However, for these configurations, no feasible solution was found with 15 rounds, thus not taken into account.
- When T =2 s and T = 1.6 s (respectively T = 1.2 s) for a number of rounds equal to 10 (respectively 5), we can notice that AIF-P deploys a lower number of AIFs than AIF-P-var, however the distributed learning is lower with AIF-P. This can be explained by the fact that this latter uses hardware accelerators with the few number of deployed AIFs.
- Figures 5.7b and 5.7c show that the baseline algorithm and AIF-P-nohw cannot find any feasible placement solution when having more than 5 rounds. This can be explained by the increased number of exchanges between the AIFs and the FL server which has a direct impact on the overall learning time. In this case, finding a feasible solution with respect to the imposed target times is not possible. On the other hand, AIF-P and AIF-P-var show higher performance in

placing the AIFs with stringent time constraints thanks to the use of hardware accelerators.

- In some cases, even though the number of available nodes that can be used to deploy the necessary number of AIFs w.r.t the target learning time is sufficient, we can notice that, the algorithms cannot produce any feasible solution. In fact, after each round the AIF should exchange the local model parameters with the FL server as our solution takes into consideration the communication latency in the overall learning time (Section 5.2.2. Even if the communication delay is insignificant to the training time when performing small number of rounds, it is not negligible any longer if the desired training quality is higher, hence using more rounds.

- It is worth mentioning that AIF-P-U yields different solutions for each simulation, i.e., when the placement and the number of accelerators change, which explains the high variance represented by the error bars. In fact, this approach deploys as many AIFs as the number of installed accelerators regardless of the number of rounds and the target learning time.

- Although the algorithm AIF-P-U does not lead to globally performing solutions, this type of algorithms can be suitable for cases where it is not acceptable not to place AIFs while tolerating marginal violations on the target learning time.

- It is worth mentioning that there exist some configurations where AIF-P and AIF-P-var do not provide feasible solutions, this happens when we have strict time constraints. For instance, through the 15 instances, both algorithms provide 30% of unfeasible solution with T = 2 s and $R = 15$. This number increases when increasing the number of rounds to 20. This is related to the placement and the number of hardware accelerators on network nodes provided for each simulation.

As we can see from the performance of approaches using hardware accelerator, this latter reduces the per-round learning time, hence offering more flexibility on placing and activating AIFs and consequently training the model for a higher number of rounds.

5.5 Conclusion

In this chapter, we tackled the problem of artificial intelligence function placement in a federated learning environment where hardware accelerators can be used to increase the learning time efficiency.

We proposed a MILP model that takes into consideration several challenges, mainly the location of FL nodes and the communication and processing delays. We then proposed two different variants of the model and evaluated the performance of the proposed solutions while comparing them to a baseline placement solution using a greedy algorithm.

We have shown that even if the greedy algorithm performs better performance in scaling where it has the lowest execution time, it has inferior performance when compared to AIF-P, AIF-P-var and AIF-P-U. On the other side, AIF-P and AIF-P-var show similar performance in finding feasible solutions but with different behaviors, i.e., the former reduces the learning time and increases the number of active AIFs while the latter behaves the opposite.

Choosing between these two algorithms can be based on time constraints and the overall quality of learning, which decreases if data is highly distributed, preventing the local model from having a sufficient view of the system. AIF-P-U is able to provide feasible placement solutions based on a violation rate, this approach shows its benefit when unfeasible solutions are not acceptable.

The obtained results shows that the proposed models allow to increase the number of rounds by 200% (respectively 300%) w.r.t to a target time equal to 2 s and 1.2 s (respectively 1.6 s) when compared to the baseline placement solution thanks to hardware acceleration.

An important finding in the placement of distributed AIF is that strong statement on hw accelerators and/or influence in the placement. In the next chapter, we will focus on mitigating the straggler effect in FL while making use of HWAs.

Chapter 6

In-network Federated Learning Control Scheme

Contents

6.1 Introduction

In Chapter 5, we presented a baseline formulation for the placement of the FL AIF participants to increase the training efficiency. In this chapter, we rely on the aforementioned problem while considering synchronization management, one of the most important challenges in FL. Indeed, in FL model updates arrival at the server from multiple edge AIFs can suffer from a so strong desynchronization that data may no longer be valuable, hence it may get discarded by the FL server for retraining: these nodes are called stragglers.

The main reason behind the appearance of stragglers is system and data heterogeneity [83] [53]. System heterogeneity is related to computing capacity (i.e., high computation delays) and channel conditions (i.e., high communication delays). On the other hand, data heterogeneity refers to the case where data is not independent and not identically distributed (non-iid) among FL participants. Stragglers slow down the learning process as the aggregation task at the FL server is only triggered once all the local parameters are received resulting in long convergence time to achieve the required accuracy. Mitigating stragglers consists of reducing the overall training time to achieve a desired accuracy in a timely manner. To do so, different techniques can be adopted. For example, [53] classified the works in the literature to minimize the convergence time into three categories: (i) data distribution adjustment, to mitigate the non-iid data, (ii) model compression, to reduce both the local training time and (iii) client selection to select the FL participant with high computation and communication performance.

In this chapter, we present the IFLC (In-network Federated Learning Control), that is **an adaptive scheme for the usage of HWAs in distributed -in-network learning systems to compensate for end-to-end network and learning delays variations leading to stragglers**. In the previous chapter, we defined a preliminary FL-AIF placement framework, using a mathematical programming approach. In this chapter, we go beyond the existing work, reformulating the model to control stragglers, defining a refined end-to-end training latency modeling, and proposing a polynomial optimal resolution algorithm hence supporting near-real-time orchestration of FL-AIFs.

The contributions of this work can be summarized as follows[1]:

- We present an original end-to-end learning latency model jointly considering learning and

[1]This work is submitted for publication in the journal of IEEE *Transactions on Network and Service Management.*

communication delays. Also, we consider a deterministic scenario where the estimated end-to-end training time of the selected clients is within the imposed time limit, and stochastic scenarios where the selected FL clients may generate additional delays resulting in stragglers;

- Based on the FL-based architecture (Figure 5.2, Chapter 5), we formulate a joint optimization problem of FL server and FL clients placement, and FL straggler minimization using a polynomial algorithm for resolution in addition to a MILP (Mixed-Integer Linear Programming) formulation;
- We evaluate the IFLC using a real-world FL-based anomaly detection [12] in order to assess how well the proposed approach can (i) increase the local training time efficiency, (ii) minimize the occurrences of FL stragglers and (iii) find a trade-off between the number of active FL clients and CPU utilization;
- We compare our approach to a first-fit algorithm. Numerical results show that our algorithm helps increase the number of selected FL clients that positively contribute to the learning task by up to 100%.

6.2 Problem Statement

We first describe the AIF network delay model, the addressed problem and the proposed resolution algorithm. We refer to our framework as In-network Federated Learning Control (IFLC) to express the fact that we aim at controlling the latency phenomena arising in in-network FL applications subjected to stringent training model update targets[2]. Note that in this work, only system heterogeneity is considered. We do not consider additional delays related to non-iid data and the time needed to receive data at the edge AIFs is considered negligible as well.

The notations used are summarized in Table 6.1.

We consider a FL-based AIF system (Figure 5.2) composed of a set N of AIFs running on physical edge servers with heterogeneous CPU resources. Let c_i be the number of available CPU cores on edge

[2]It is worth noting that placing an AIF can mean copying a function image to a physical node or selecting a pre-fetched AIF already installed in the physical node, so that its actual instantiation can be a near-real-time operation in a similar time-scale to that of near-real-time execution algorithms.

Sets and parameters	
N	set of edge physical servers, with $n = \|N\|$.
N_s	set of physical servers for FL server placement.
c_i	number of available CPU cores on node i
p_{ik}	training time for $i \in N$ with k active edge AIFs.
τ	target distributed learning time (one FL round).
d_{ij}	communication latency between node i and node j.
α_{ik}	≥ 1, acceleration factor at node i with k active edge AIF
h_i	assumes value 1 if a HWA is available on node i
H	maximum number of allowed HWAs.
$\mathcal{S}$	set of delay scenarios.
q_s	probability of scenario s.
β_{ik}^s	drift of the learning time on scenario s on node i with k active edge AIFs.
η_{ij}^s	drift of the propagation delay on scenario s from node i to node j.
Δ	maximum tolerated end-to-end delay.

Table 6.1: Notations.

node i. We denote by N_s the set of physical servers used for the AIF server placement. It is worth mentioning that $N_s \equiv N$ if the FL server AIF can be installed on the edge servers and $N_s \cap N = \emptyset$ otherwise.

Additionally, we consider a set of FL client AIFs that can be deployed on top of each physical node to run a given FL-based application. An AIF receives data streams from external nodes, triggering the training task. We consider that an AIF consumes all CPU resources that are made available to it on the physical node[3].

We suppose that each physical node can be equipped with HWA to increase the local training efficiency. We denote by α_{ik} the acceleration factor at node i when k FL client AIFs are active. The total number of HWAs is limited by a constant H. In the following, we present the modeling of the end-to-end learning time.

6.2.1 End-to-end training time modeling

The global training time needed to update the AI model at the server AIF is therefore a function of different training and propagation times and the number of edge AIFs. Figure 6.1 depicts the training time model components.

Definition 1 (Local Training Time - p)**.** Let p_{ik} be the local training time of an AIF on node i when k AIFs are active.

This parameter depends on (i) the number of available CPU cores at the physical layer, (ii) the enabling of HWAs, and (iii) the amount of training data. More precisely, the training time is directly related to the data size, e.g., the training time in neural networks is evaluated by the number of floating point operations which depends on the data size and the architecture of the neural network [84]. Consequently, the local training time reduces with the number of active AIFs as data is distributed amongst them. We define p as an upper bound for the overall training time during a given round.

Definition 2 (Propagation Delay - d)**.** Let d_{ij} be the communication latency on the link that interconnects nodes i and j.

Transmission delays, negligible due to small volume of data exchanged (if2, Figure 5.2), could also be incorporated in d.

[3]This corresponds to the default behavior of container-based services.

Definition 3 (End-to-End (E2E) Training Time - χ)**.** Let χ_{ik} be the sum of the local training time at node i (while k AIFs are active) and the propagation delay between the nodes deploying this client AIF and the FL server AIF j.

As in real world scenarios stochastic delays may apply on end-to-end training times, we consider two different behaviors for an AIF: a deterministic behavior where the local training time and the propagation delay are the same as expected, and a stochastic behavior where additional delays may apply to the local training time, to the propagation delay or both.

Definition 4 (Training Time Drift - β)**.** Let β be the stochastic delays applied to the local training time.

This parameter is supposed to be unknown, even if it can be empirically characterized from real systems. Furthermore, we define a set $\mathcal{S}$ of possible *scenarios* that define the intensity of the stochastic delays. In such a way, for each scenario s we have a realization of the training time drift β: β_{ik}^{s} for each node i and k deployed AIF.

Definition 5 (Propagation Delay Drift - η)**.** Let η be an additional random value applied to the propagation delay (e.g., it can be traffic dependant following a queuing mode or traffic independent where it is influenced by the link length).

For each scenario s a drift is applied to the propagation delay η is : η_{ij}^{s} for each pair of edges i and j.

Definition 6 (Target Time - τ)**.** Let τ be the target learning time after which aggregation is triggered at the server AIF.

Definition 7 (Maximum Tolerated Delay - Δ)**.** Let Δ be the tolerated elapsed training delay between the last accepted reception of the parameters from an edge AIF, and the effective start of the parameters aggregation at the server AIF.

Definition 8 (AIF Straggler or Straggling AIF)**.** An AIF straggler or straggling AIF i is a client AIF whose E2E training time χ is greater than the threshold but remains within the tolerated additional time Δ (i.e., $\tau \leq \chi \leq \tau + \Delta$).

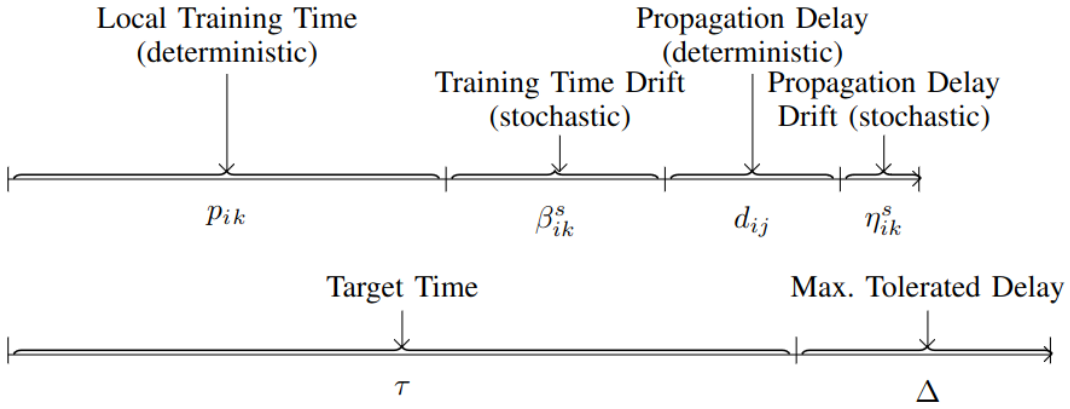


Figure 6.1: E2E training time (χ) model components and threshold.

Note: parameters sent by an AIF straggler are aggregated by the server AIF if $\chi \leq \tau + \Delta$. The values of τ and Δ depends on the use case specification requirements, with $\Delta < \tau$.

We consider two possible settings for the FL AIFs:

- 'edge-edge': both server and edge AIFs are running at the edge nodes: the propagation delays in that case can be expected to be negligible as compared to the training time, considering the possible duplication of the FL server instance close to the clients.
- 'core-edge': the server is placed at a core location (beyond aggregation nodes) whereas edge AIFs are placed at the edges: the propagation delays for this setting can be expected to be of the same magnitude as the training time.

6.2.2 In-Network Federated Learning Control: IFLC

Under the delay model, we can define the IFLC problem as follows. Given a set of physical nodes N and a defined target time for a specific application, find an optimal placement of the FL server AIF and the FL client AIFs to ensure that:

- the maximum E2E training time does not exceed the time requirements imposed by the application,
- HWA, if available, can be enabled to reduce the local training time,
- the CPU utilization is minimized,

- the average number of AIFs exceeding the target time is minimized, including the stochastic scenarios (see definitions 4 and 5).

Indirectly, the number of active AIFs is pushed down towards optimality;

6.3 Mathematical Model

In the following, we present the mathematical programming model that corresponds to the IFLC scheme. We use a set of binary variables: x_i represents the activation state of the AIF on node i, hence it takes value 1 if node i is used to deploy an AIF. y_j provides the position of the FL server AIF, if it is equal to 1 then the FL server is placed on node j. If an AIF is installed on node i and the FL server is placed on node j then ξ_{ij} is equal to 1. ζ_{ik} and z_k are used to count the total number of active AIF. ψ_{ik} is equal to 1 if the hardware accelerator is present and used on node i and k AIFs are active.

On the other hand, real variables are introduced. t_i represents the local training time when an AIF is active on node i, δ_{ik} represents the amount of reduction in the local training time due to hardware acceleration, when k AIFs are active on node i and χ_i represents E2E training time of the client AIF deployed on node i. Finally, we introduce the real variables $\tilde{t_i}^s$ and $\tilde{\delta_{ik}}^s$ which correspond to the stochastic local training time and the reduction in the stochastic local training time, respectively, for scenario s. $\tilde{\chi_i}^s$ is the E2E training time of the client AIF deployed on node i with scenario s

The mathematical notations are summarized in 6.2.

6.3.1 Core model constraints

6.3.1.1 FL clients and FL server AIFs placement

We need to determine the location of the FL server AIF and the number and location of the client AIFs in order to guarantee that each AIF can train and send the model parameters to the FL server on time.

Constraints (6.1) impose that the FL server AIF is installed on one and only one node. (6.2) impose that the node that hosts the server cannot host an edge client AIF.

We recall that variable ξ_{ij} is used to represent the fact that a FL client AIF is installed on node i

Binary variables	
x_i	1, if a client AIF is active on node i
y_j	1, if the FL server AIF is installed on node j
w_i	1, if the hardware accelerator is activated on node i
ζ_{ik}	1, if an AIF is active on node i with k deployed AIFs
ξ_{ij}	1, if a client AIF is installed on node i and the FL server AIF is installed on node j
σ_i^s	1 if a client AIF on node i is a straggler in scenario s
ψ_{ik}	1, if a hardware accelerator is present and used on node i and k AIFs are active.
z_k	total number of activate AIF
Continuous variables	
$\tilde{t_i}^s$	distributed training time on node i
χ_i	E2E training time of the client AIF deployed on node i when k AIFs are active
δ_{ik}	time reduction at node i when k AIFs are active
$\tilde{\delta_{ik}}^s$	time reduction at node i when k AIFs are active
$\tilde{\chi_i}^s$	E2E training time of the client AIF employed on node i when k AIFs are active with scenario s
$\tilde{\delta_{ik}}^s$	time reduction at node i when k AIFs are active

Table 6.2: Mathematical notations.

and the FL server is installed on node j. Therefore, when $\xi_{ij} = 1$ the AIF installed on node i yields a communication latency of d_{ij}. Constraints (6.3) and (6.4) are consistency constraints. If the server is not installed on node j then all the variables ξ_{ij} must be equal to zero. For a given node i, one and only one variable ξ_{ij} can assume value 1 if a client AIF is installed on node i, otherwise they are all equal to zero.

$$\sum_{j \in N_s} y_j = 1 \tag{6.1}$$

$$y_i + x_i \leq 1 \qquad \forall i \in N \tag{6.2}$$

$$\xi_{ij} \leq y_j \qquad \forall i \in N, j \in N_s \tag{6.3}$$

$$\sum_{j \in N_s} \xi_{ij} = x_i \qquad \forall i \in N \tag{6.4}$$

6.3.1.2 Training time characterization

Constraints (6.5) allow to calculate the deterministic training time of node i when k AIFs are deployed.

Constraints (6.6) and (6.7) are consistency constraints, that is, when variables z_k or x_i are null, the corresponding ζ_{ik}, t_i variables for node i are also null.

Constraints (6.8) together with constraints (6.9) allow us to count the number of deployed AIFs.

$$t_i = \sum_{k=2}^{n} p_{ik} \zeta_{ik} \qquad \forall i \in N \tag{6.5}$$

$$\sum_{k=2}^{n} \zeta_{ik} = x_i \qquad \forall i \in N \tag{6.6}$$

$$\sum_{i \in N} \zeta_{ik} \leq |N| z_k \qquad \forall k \in 2..n \tag{6.7}$$

$$\sum_{k=2}^{n} z_k = 1 \tag{6.8}$$

$$\sum_{k=2}^{n} k z_k = \sum_{i \in N} x_i \tag{6.9}$$

6.3.1.3 Hardware acceleration for deterministic training

We allow the use of hardware accelerators to reduce the training time. We introduce the necessary constraints to evaluate the impact of the hardware accelerators on the local training time. Constraints (6.10) impose that the hardware accelerator can be used only if available. Constraints (6.11) impose that a maximum number H of hardware accelerators can be used.

$$w_i \leq h_i \qquad \forall i \in N \tag{6.10}$$

$$\sum_{i \in N} w_i \leq H \tag{6.11}$$

Constraints (6.12)-(6.15) allow to evaluate the gain in the deterministic local training time obtained using hardware acceleration while associating the two variables δ_{ik} and w_i to keep consistency.

$$\psi_{ik} \leq \frac{\zeta_{ik} + w_i}{2} \qquad \forall i \in N, k \in 2..n \tag{6.12}$$

$$\delta_{ik} \leq \psi_{ik} \left(1 - \frac{1}{\alpha_{ik}}\right) p \qquad \forall i \in N, k \in 2..n \tag{6.13}$$

$$\delta_{ik} \leq \left(1 - \frac{1}{\alpha_{ik}}\right) (p_{ik}\zeta_{ik}) \qquad \forall i \in N, k \in 2..n \tag{6.14}$$

$$\delta_{ik} \geq \left(1 - \frac{1}{\alpha_{ik}}\right) (p_{ik}\zeta_{ik}) - \quad (1 - \psi_{ik})p \qquad \forall i \in N, k \in 2..n \tag{6.15}$$

Note that p represents the maximum deterministic training time and can be calculated as follows:

$$p = \max_{k=2..n, i \in N} p_{ik} \tag{6.16}$$

6.3.1.4 Target time

For each node i, constraints (6.17) compute the E2E training time of an active AIF on node i. Constraints (6.18) ensure that the maximum E2E training time that an active AIF can achieve does not exceed the accepted target time τ. These constraints are always valid even when no AIF is installed. In fact, variables t_i and δ_{ik} assume value zero when $x_i = 0$ (see, constraints (6.4), (6.5)-(6.6), and (6.14)-(6.15)).

$$\chi_i = t_i - \sum_{k \in 2..n} \delta_{ik} + \sum_{j \in N_s} d_{ij}\xi_{ij} \qquad \forall i \in N \tag{6.17}$$

$$\chi_i \leq \tau \qquad \forall i \in N \tag{6.18}$$

6.3.2 Stochastic variant

In the following, we introduce the stochastic variant of the AIF placement model. The goal is to introduce robustness against different realization scenarios. We add the following constraints to the aforementioned model.

Constraints (6.19) calculate the stochastic local training time for node i when k AIFs are active and a delay β_{ik}^s is applied.

$$\widetilde{t_i}^s = \sum_{k=2}^{n} (p_{ik} + \beta_{ik}^s)\zeta_{ik} \qquad \forall i \in N, s \in \mathcal{S} \tag{6.19}$$

In the same way as in deterministic model, constraints (6.20)-(6.22) allow to evaluate the gain in local training time obtained using hardware acceleration while considering the additional delays applied to the training time.

$$\widetilde{\delta}_{ik}^s \leq \psi_{ik}\left(1 - \frac{1}{\alpha_{ik}}\right)\widetilde{p} \qquad \forall i \in N, k \in 2..n, \forall s \in \mathcal{S} \tag{6.20}$$

$$\widetilde{\delta}_{ik}^s \leq \left(1 - \frac{1}{\alpha_{ik}}\right)(p_{ik} + \beta_{ik}^s)\zeta_{ik} \qquad \forall i \in N, k \in 2..n, s \in \mathcal{S} \tag{6.21}$$

$$\widetilde{\delta}_{ik}^s \geq \left(1 - \frac{1}{\alpha_{ik}}\right)(p_{ik} + \beta_{ik}^s)\zeta_{ik} - (1 - \psi_{ik})\widetilde{p} \qquad \forall i \in N, k \in 2..n, \forall s \in \mathcal{S} \tag{6.22}$$

Note that $\widetilde{p}$ represents the maximum stochastic training time and can be calculated as follows:

$$\widetilde{p} = \max_{k=2..n, i \in N, s \in S} (p_{ik} + \beta_{ik}^s) \tag{6.23}$$

Finally, constraints (6.24) compute the E2E stochastic time of node i and constraints (6.25) impose that the E2E training time of an active AIF, including the additional delays applied to both the training and propagation times, are below the threshold. It is worth noticing that in this case we accept a maximal response time $\tau + \Delta$ with the aim of reducing the number of stragglers.
Also, these constraints are always valid even when no AIF is installed where $\widetilde{t_i}^s$ and $\widetilde{\delta}_{ik}^s$ assume value zero when $x_i = 0$ (see, constraints (6.4), (6.5)-(6.6), and (6.21)-(6.22)).

$$\widetilde{\chi_i}^s = \widetilde{t_i}^s - \sum_{k=2}^{n} \widetilde{\delta}_{ik}^s + \sum_{j \in A} (d_{ij} + \eta_{ij}^s)\xi_{ij} \qquad \forall i \in N, s \in \mathcal{S} \tag{6.24}$$

$$\widetilde{\chi_i}^s \leq \tau + \Delta\sigma_i^s \qquad \forall i \in N, s \in \mathcal{S} \tag{6.25}$$

6.3.3 Objectives

The goal is to minimize the number of AIFs that can be in a straggling situation in order to guarantee a certain level of performance of the learning process with minimum costs. To this end, we introduce two objectives. We first minimize a measure of the stragglers in the system and then, we search for solutions with the minimal utilization of resources.

6.3.3.1 Minimizing the number of stragglers

(6.26) allow to minimize the expected number of stragglers associated with a probability for each scenario q_s.

$$\min \sum_{s \in \mathcal{S}} q_s \sum_{i \in N} \sigma_i^s \tag{6.26}$$

6.3.3.2 Minimizing the number of computational resources

After determining the optimal solution of the previous problem, we can further minimize the number of computational resources while limiting the increase of the expected number of stragglers. We denote by $\sigma^\star$ the minimum number of AIF in a straggling situation provided by (6.26). We formulate our second phase optimization problem as:

$$\min \sum_{i \in N} c_i x_i \tag{6.27}$$

We introduce the additional constraints (6.28) to limit the increase of the number of stragglers.

$$\sum_{s \in \mathcal{S}} q_s \sum_{i \in N} \sigma_i^s \leq \sigma^\star + \epsilon \tag{6.28}$$

Note that the objective in (6.26) along with constraints (6.28) are only used by the stochastic variant.

Additionally, we introduce the following domain constraints to complete the model:

$$t_i \geq 0 \qquad \forall i \in N \tag{6.29}$$

$$\widetilde{t_i}^s \geq 0 \qquad \forall i \in N, s \in S \tag{6.30}$$

$$\delta_{ik} \geq 0 \qquad \forall i \in N, k \in 2..n \tag{6.31}$$

$$\widetilde{\delta_{ik}}^s \geq 0 \qquad \forall i \in N, k \in 2..n, s \in S \tag{6.32}$$

$$x_i, w_i, y_j, \xi_{ij} \in \{0, 1\} \qquad \forall i \in N, j \in N_s \tag{6.33}$$

$$z_k \in Z \qquad \forall k \in 2..n \tag{6.34}$$

$$\zeta_{ik}, \in \{0, 1\} \qquad \forall i \in N, k \in 2..n \tag{6.35}$$

6.4 Resolution algorithm

We propose an IFLC resolution algorithm for the deterministic case where the structure is based on the following observations:

- the possible locations of the FL server AIF is given by $|N_s|$ (or $|N|$ for the edge-edge setting)
- the number of installed AIFs is limited by $|N|$ (or $|N| - 1$ for the edge-edge setting),
- for each server location and given number of installed AIFs, χ can be pre-calculated.

The proposed algorithm search for the best solution for each given combination (j, k) of AIF server location (j) and number of installed edge AIFs k and keep the best one, see Algorithm 1. In the edge-edge setting, the server is chosen from the set N and its location is removed from the available AIFs location. The decision of activating an AIF is done working on the set of locations ordered by increasing c_i, breaking ties using χ (smallest first). Note that differently than the mathematical

model presented in the previous section, this algorithm considers that the end-to-end training time component χ as a parameter, computed for each combination (j, k) of AIF server location (j) and number of installed edge AIFs k.

Algorithm 1: General IFLC Scheme

output : $S^\star$: set of active edge AIFs, j: server AIF position

```
best_cost = ∞
S* = ∅
for j ∈ N_s do
    for k ∈ 1..n do
        Ñ ← available nodes in decreasing order of c_i
        (feasible, S, cost) = BEST_PLACEMENT(k,j,Ñ)
        if cost ≤ best_cost and feasible then
            bestcost = cost
            FL_server = j
            S* = S
return (S*, FL_server)
```

To allow a compact representation of the placement procedure, we report here the calculation of χ for a given couple (j, k). The deterministic E2E training time for a given node i can be calculated as:

$$\chi_i = p_{ik} + d_{ij} \tag{6.36}$$

and when the hardware accelerator is installed:

$$\chi_i^{ha} = p_{ik} - \delta_{ik} + d_{ij} \tag{6.37}$$

Note that if $\chi > \tau$, the edge AIF cannot be placed on node i, regardless of whether HWA is available or not. Therefore, the BEST_PLACEMENT procedure inspects the list of ordered nodes $\tilde{N}$ and checks whether is possible to place an edge AIF without exceeding the threshold sequentially. HWAs are used (if available) only if necessary to reduce χ under the threshold τ. BEST_PLACEMENT is presented in Algorithm 2[4]. For ease of presentation, we do not explicitly add the line that save the location of the HWA (assumed saved by the procedure). Also, note that we only present the placement procedure for the deterministic case.

We can observe that at each step of the function presented in Algorithm 2, the less expensive AIF location for which the E2E training time is below the threshold is selected. If at the end of the

[4]Where no HWA is available the parameter H assumes value zero.

Algorithm 2: Best placement - deterministic

input : j: server location, k number of active edge AIFs,
$\tilde{N}$ set of available nodes for locating the AIFs
output : feasible: bool, S: set of active edge AIFs,
cost: solution cost

kcount = 0, cost = 0, hwa = 0, $S = \emptyset$
Calculate deterministic E2E training times
`/* try to locate` k `AIFs */`
while $\tilde{N} \neq \emptyset$ *and kcount* $< k$ **do**
  i=pop($\tilde{N}$)
  if $\chi_i \leq \tau$ **then**
    kcount ++, cost += c_i, $S = S \cup \{i\}$
  else if $\chi_i^{ha} \leq \tau$ *and hwa* $< H$ **then**
    kcount ++, cost += c_i, hwa ++, $S = S \cup \{i\}$
if *kcount* == k **then**
  return (True, S, cost)
else
  return (False, $\emptyset$, ∞)

process, the number of selected edge AIFs is lower than k, it means that k AIFs do not allow a viable E2E training time (with the number of available HWAs).

6.4.1 Time and space complexity

The algorithm repeats the procedure "BEST_PLACEMENT()" for each couple (j, k) of FL server location and number of instantiated AIFs, i.e. $|N||N_c|$ iterations. In addition, the placement procedure performs a sorting of the AIF locations and an inspection of the resulting list. Thus, the overall time complexity for the algorithm is of the order of $O(|N||N_c|\log(|N|))$, $\approx O(n^2\log(n))$, where $n = |N|$ and under the assumption that $|N_c| = O(n)$.

6.4.2 FIRST-FIT algorithm

As a lowest-complexity benchmark, we introduce a baseline placement algorithm to compare with IFLC strategies. It is a first-fit algorithm, of $\approx O(n\log(n))$ time and space complexity, that prioritizes the nodes with the highest CPU resources where it increases the number of deployed AIFs until there is no more decreasing in the E2E training time. FIRST-FIT is given in Algorithm 15.

Algorithm 3: FIRST-FIT algorithm

output : S: set of active edge AIF, j: server AIF position

$\tilde{N} \leftarrow$ sort N in decreasing order of CPU resources
$S \leftarrow 0$
$E2E\_time = 0$
$E2E\_time\_final = \infty$
$k = 0$, *number of AIFs*
$j = random(\tilde{N})$, $\tilde{N} = \tilde{N} \setminus \{j\}$
while $\tilde{N} \neq \emptyset$ **do**
    i=pop($\tilde{N}$)
    $S = S \cup i$, $k+ = 1$
    Update($E2E\_time$, k)
    if $E2E\_time\_final \leq E2E\_time$ **then**
        $S = S - \{i\}$, $k- = 1$
        return (S)
    Update($E2E\_time\_final$, k)
return (S, j)

6.5 Simulated Instances

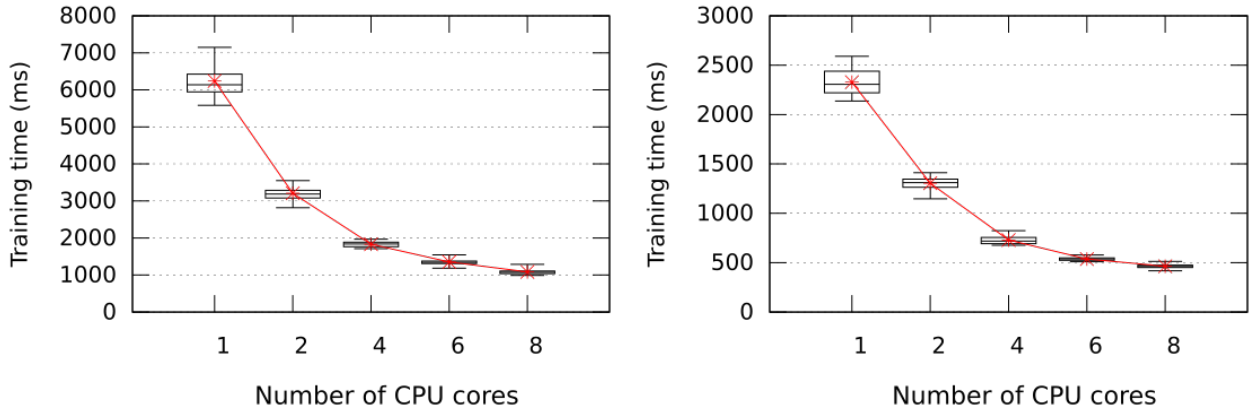
We report how we set-up the numerical evaluation environment, including simulated scenario and the used dataset.

6.5.1 AIF application

In order to evaluate IFLC strategies, we consider the FL-based framework proposed in [12] where each AIF is an implementation of an LSTM autoencoder neural network system to detect anomalies in a 5G stack. The goal of this framework is to detect anomalies at different system levels, i.e., physical level, virtual/container level and access level, using thousands of time-series issued by probes from network functions, physical servers, Eth/IP and radio links. Probes are collected from a 5G testbed replaying traffic traces of a European operator, from the ANR COCO5G project (`https://coco5g.roc.cnam.fr`), in the Lozere region in France for 3 months in 2019.

The data collected from the probes form the 5G3E dataset from [80] provides few dozens of feature time-series for each resource group, where groups are related to CPU, RAM, storage and link states. We use data batches of 4000 samples to train the aforementioned AIML model: it corresponds to 10

minutes of collected data, assumed to be the retraining time of the system and could vary in general depending on the sampling rate. The batch size is set to the data size, hence considering all the samples. The number of epochs (E) is 10 and the model is trained for one round ($R = 1$). The data is then evenly load-balanced as a function of the number of edge AIFs employed. The rest of the hyper-parameters are set as explained in [12].



(a) 2 AIFs.

(b) 6 AIFs.

Figure 6.2: Training time distribution in function of number of active edge AIFs and available CPU cores. $R = 1$, $E = 10$.

6.5.2 Computation of training and propagation time samples

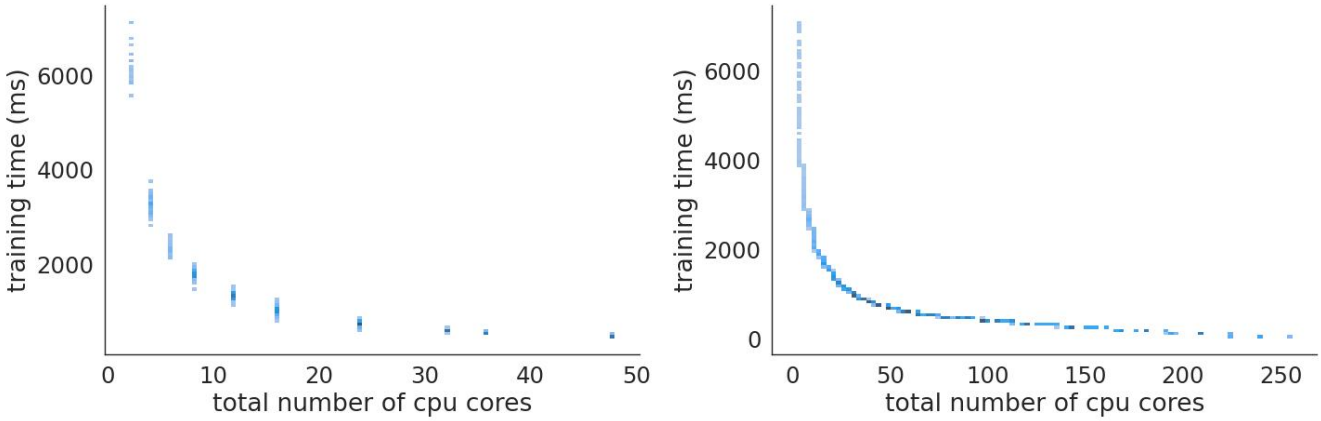
We generate the training time samples as a function of the number of edge AIFs and the amount of computation resources using the aforementioned AIF application. Figure 6.2 depicts the distribution of the maximum local training time for different numbers of edge AIFs and CPU cores. We can remark that the training time decreases with the increase of the number of AIFs and of available CPU cores up to a certain threshold.

In contrast to conventional user-device FL-based services, this framework considers an in-network service where the time scale at which the anomaly detection model is expected to react is on the order of few seconds, or even sub-second. Nonetheless, it is worth mentioning that our IFLC scheme can be applied on any FL application.

For the purpose of evaluating IFLC on large instances, we generate a synthetic set of pseudo-random training times that approximate a pre-specified correlation coefficient between the training time values, the number of active AIFs and the number of available CPU cores. This correlation coefficient is

retrieved from the original data. We make available the samples and related scripts for the simulations in [85]. Note that the generated dataset contains training times for different number of active AIFs and available CPU cores.

Figures 6.3 depict the distribution of the training times of both the original and synthetic datasets, based on the total number of CPU cores (i.e., number of CPU cores that are used by all active AIFs).



(a) original dataset (b) synthetic dataset

Figure 6.3: Training time distribution vs the number of CPU cores.

As a function of the AIF positioning setting, we configure the E2E training time components as follows:

- 'edge-edge' setting: we define the maximum one-way latency between the furthest edge AIF and the server AIF as the 25% quantile value of the training time during one epoch divided by 10.
- 'core-edge' setting: the highest value of the one-way latency between the furthest FL client AIF and the FL server is set equal to the mean value of the training times during 10 epochs.

We consider a combination of deterministic and stochastic behaviors for both propagation and training times as shown in Table 6.3. Note that 'S-D'case is not cited because it generates the same solutions as 'S-S'. This can be explained by the fact that both cases apply stochastic delays on the local training time, and since the latter has the highest impact on the E2E training time compared to the propagation delay, the placement solution is the same for both cases as they have similar strictness on time constraints. We set the stochastic drifts as proportional to the nominal values of the training time and propagation delays, respectively.

cases	training time	propagation delay
case D-D	deterministic	deterministic
case D-S	deterministic	stochastic
case S-S	stochastic	stochastic

Table 6.3: Training time and propagation delays options.

6.5.3 Simulated network instances

We use the Mandala topology already presented in Chapter 5. Also, we consider that each node can be equipped with 1, 2, 4, 8 or 16 CPU cores.

We compare four resolution approaches:

- *no-HWA*: degenerate IFLC with no HWAs employed.
- *IFLC-8*: 50% of edge nodes (i.e. 8) equipped with HWAs.
- *IFLC-16*: all edge nodes (i.e. 16) equipped with HWAs.
- *FIRST-FIT* baseline Alg. 6.4.2 (not enabling HWA).

The comparison is done looking at the following features:

- the number of straggling AIFs,
- the E2E training time and its variance,
- the computational overhead related to the number of active AIFs and of CPU cores.

We rely on [66] and [67] to define the acceleration factor of HWAs: accordingly, we consider that it depends on the number of active AIFs (i.e., data size) and the number of available CPU cores (i.e., number of threads). More precisely, the experimental evaluation in [66] have shown that the acceleration factor decreases by 370% when increasing the number of CPU cores from 1 to 16. Based on this evaluation, we applied a piece-wise linear fitting function [86] to retrieve the acceleration factor for 2, 4 and 8 CPU cores (see Figure 6.4). We repeated the same operation to generate the acceleration factors for different numbers of AIFs.

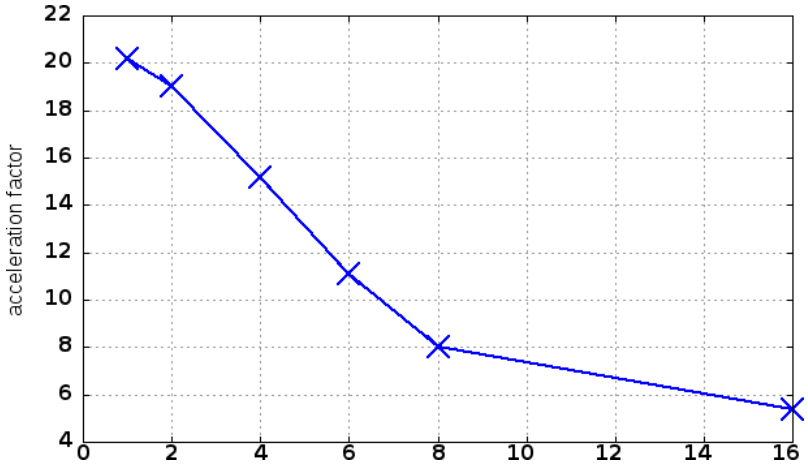


Figure 6.4: Acceleration factor: data extrapolation.

Moreover, in order to test different levels of strictness on the training time target constraints, we use different values for the target time τ (i.e., 2 and 4 s) and the number of epochs (i.e., from 60 to 105 with a step of 5 epochs). Considering constant the training times across different FL epochs, we generated local training times with an increasing number of epochs proportionally with the number of epochs starting with the baseline with 10 epochs.

Also, we consider that the maximum tolerated delay Δ is 4 times less than the target time, which roughly correspond to the maximum lifetime network connections that are not bulk transfers. Then, we range from loose timing constraint (e.g., $\tau = 4$ s with 60 epochs) to extremely rigorous ones (e.g., $\tau = 2$ s with 100 epochs). Also, additional stochastic delays may reach nearly twice the nominal time.

We run 30 instances for each approach and each different setting where the propagation time, the stochastic delays and the placement of hardware accelerators are randomly generated for each instance. The number of available CPU cores is fixed for all instances.

6.6 Experimental Results

In this section, we provide a detailed numerical evaluation focused on stragglers, training times and computation overhead. Overall, Table 6.4 presents the proportion of instances that lead to a feasible solution (with respect to the target delay bound).

For the two AIF placement settings, no-HWA could produce a solution only at most for 15% of the instances. This increases to 100% when IFLC is used. On the other side, all the solutions yielded

Approach	edge-edge	core-edge
no-HWA	15%	10%
IFLC-8	100%	100%
IFLC-16	100%	100%
FIRST-FIT	0%	0%

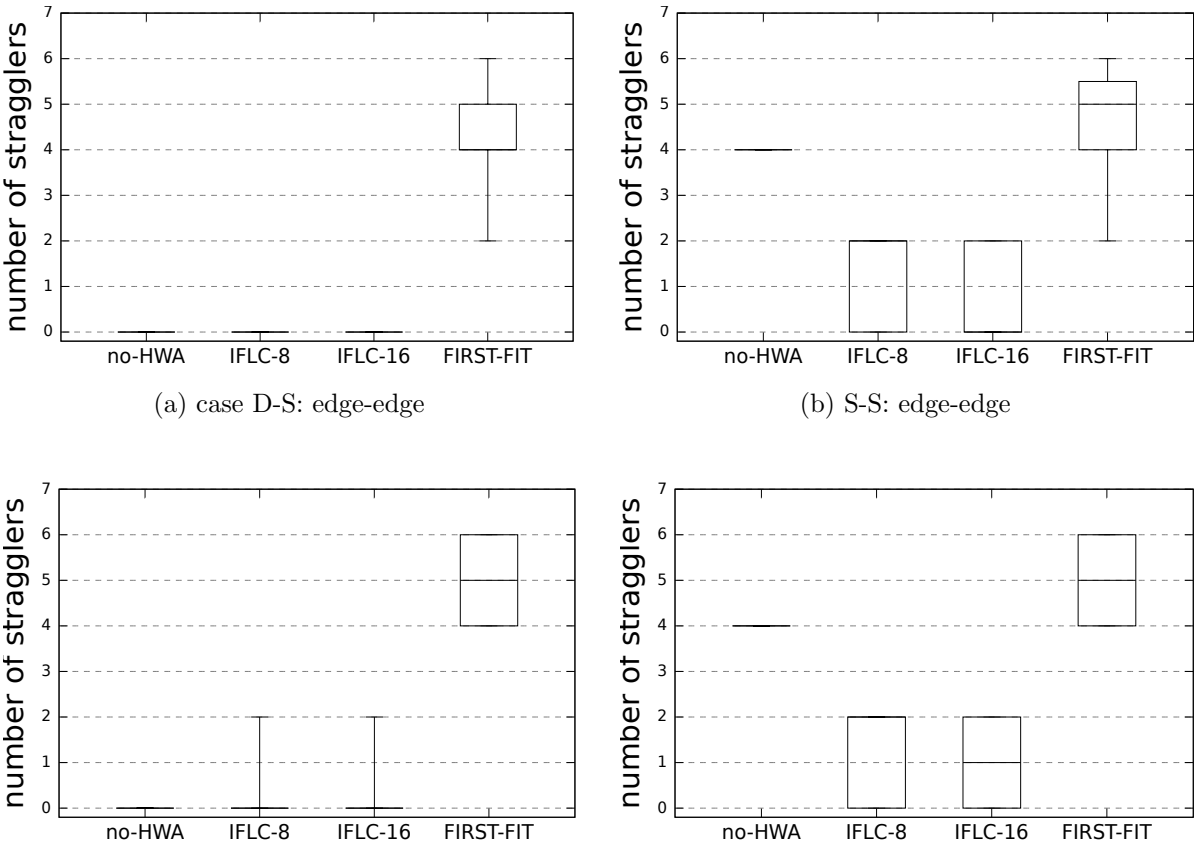
Table 6.4: Percentage of feasible instances per approach.

by FIRST-FIT algorithm are unfeasible, as it has no check on the target time. It is worth noting that if the E2E training time of an AIF exceeds the time threshold, we consider that the local training parameters cannot be used by the FL aggregation task.

6.6.1 Number of stragglers

In Figures 6.5, we present the distribution of the number of AIF stragglers for both edge and core-edge settings, and excluding the 'D-D'case since it does not model the stragglers. We can notice that:

- With IFLC, the likelihood of being in a straggling situation decreases with the number of available HWAs when the training time is stochastic. For deterministic cases, the number of stragglers is null, for all the settings.
- With FIRST-FIT, the number of stragglers is the worst, and it is higher in the core-edge setting: the placement at the edge gives more flexibility thanks to lower propagation delays, hence leading to lower E2E training times. This does not happen with IFLC, showing its robustness against high propagation delays (core-edge setting).
- For all approaches and settings, the number of straggling AIFs increase when the training time is stochastic, as it can be seen from Figures 6.5b and 6.5d. Specifically, the median value increases from 4 to 5 for FIRST-FIT with the edge-edge setting, whereas in core-edge the minimum number of stragglers increases by 2. On the other side, the highest number of stragglers generated by IFLC-8 and IFLC-16 increases from 0 to 2 for both placement settings. This number increases from 0 to 4 with no-HWA for both settings.



(a) case D-S: edge-edge

(b) S-S: edge-edge

(c) case D-S: core-edge

(d) case S-S: core-edge

Figure 6.5: Distribution of the number of straggling AIFs.

- In the 'D-S' case (see Figures 6.5a and 6.5c), our approach significantly outperforms FIRST-FIT, regardless of the number of HWAs. Under more stringent targets (S-S), FIRST-FIT yields lower number of stragglers than no-HWA with edge-edge, where the minimum number of stragglers achieved by no-HWA (i.e., 4 stragglers) corresponds to the first quartile value achieved by FIRST-FIT in the edge-edge setting, and the minimum value in the core-edge one.

Overall, thanks to time modulation, IFLC always outperforms FIRST-FIT in terms of the number of stragglers, for all the cases. HWA helps reducing the local training time which allows slow AIFs to reach lower E2E training times and consequently respect the imposed target time.

6.6.2 Training time

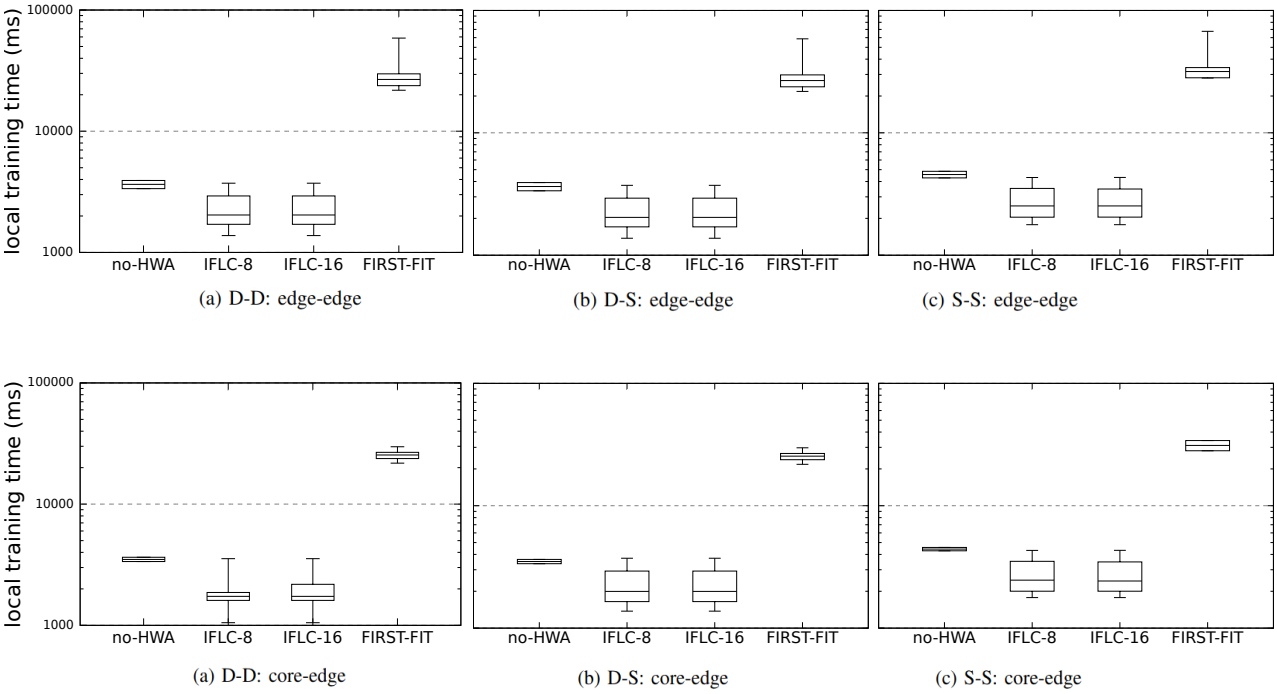


Figure 6.6: Distribution of the maximum local training time.

Figures 6.6 represent the distribution of the maximum local training times. We can notice that:

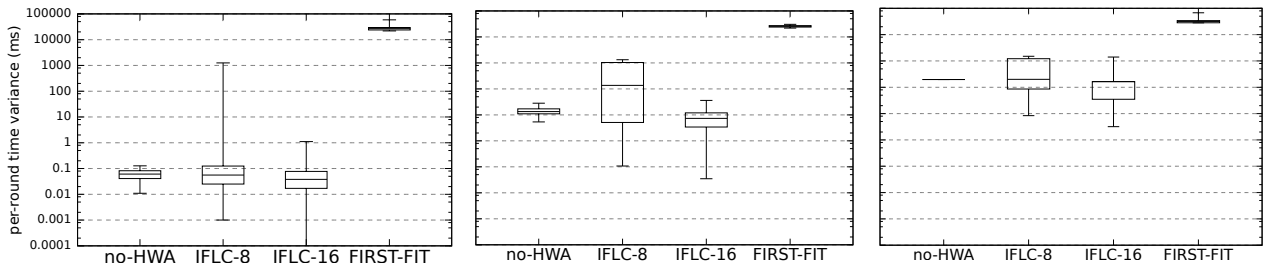
- In contrast to FIRST-FIT, IFLC approaches have similar distributions of the training time for the two placement settings. This can be explained by the fact that the latter leads to

solutions that are robust against high propagation delays. This can also be seen when comparing 'D-D' and 'D-S' cases where the training times remain the same.

- The distribution of the training time is very similar for IFLC-8 and IFLC-16 for all cases. This can be explained by the fact that both IFLC-8 and IFLC-16 yield the same placement solutions (i.e., same number of active AIFs and active HWAs) as a low number of HWAs can be sufficient to respect the target time even with very strict targets.
- Both D-D and D-S cases have lower maximum training times compared to S-S for all cases. This happens because S-S has higher local training times due to additional applied delays.

Overall, IFLC gives lowest training times thanks to HWA, followed by no-HWA and finally FIRST-FIT which yields the highest training times (which are higher than the imposed target time, hence discarded by the FL server).

Figures 6.7 report the distribution of the maximum variance in E2E training times (only for the edge-edge setting, as no major difference appears with the core-edge one). We can notice that:



(a) D-D: edge-edge (b) D-S: edge-edge (c) S-S: edge-edge

Figure 6.7: Distribution of the the variance in E2E training time.

- The lowest variance in E2E training time corresponds to IFLC-16 followed by no-HWA the IFLC-8. Indeed, IFLC deploys AIFs with close E2E training time with the aim of reducing the number of stragglers during each round, attempt favored by HWAs that get enabled to accelerate training for farthest AIFs from the server. The variance is further decreased in deterministic cases.
- IFLC-16 yields lower variance when compared to IFLC-8 for all cases. This can be explained by the fact that the former has more control in placing the AIFs on nodes with similar capacities

as HWA is present on all nodes. On the other hand, IFLC-8 has less flexibility where the node selection depends on the HWA availability. Moreover, the variance increases with time constraints as can be seen from 'D-S'and 'S-S'cases when comparing IFLC-8 and IFLC-16. In fact, IFLC-16 allows to activate HWA on nodes with similar processing capacities which results in equivalent local training times. On the other hand, IFLC-8 may not have available HWAs on these nodes and thus nodes with different processing capacities are used. This results in higher variance.

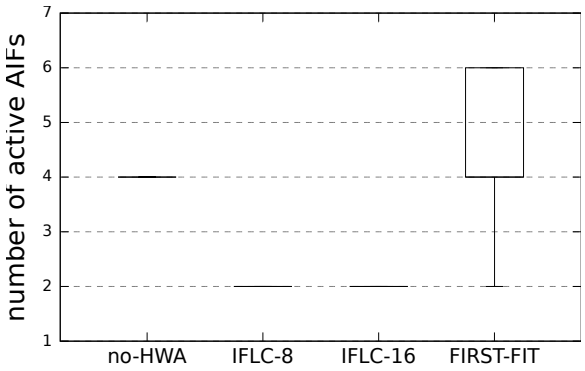
Globally, since the weight of the local training time in the E2E training time is greater compared to the propagation delay, and that training times get higher if HWA unavailability, this makes finding solutions with near local training times harder as it turns into finding nodes with higher CPU resources to respect the target time.

6.6.3 AIF computational overhead

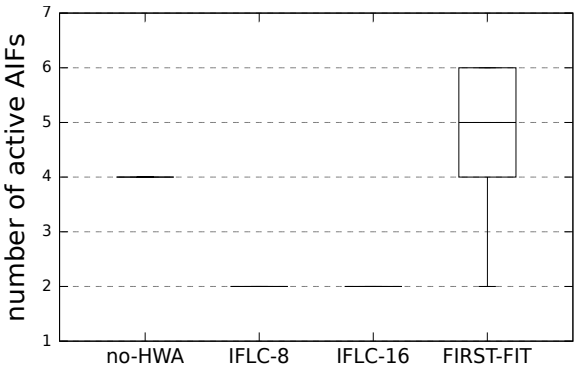
The latter observation can be clarified by Figures 6.8 and 6.9 that depict the distribution of the number of active AIFs and the total number of CPU cores that are used by the active AIFs. We only report the cases D-S and S-S in this section as D-D yields the similar distributions as 'D-S'.

Besides expectable behaviors for FIRST-FIT deriving from the previous analysis, we can highlight that:

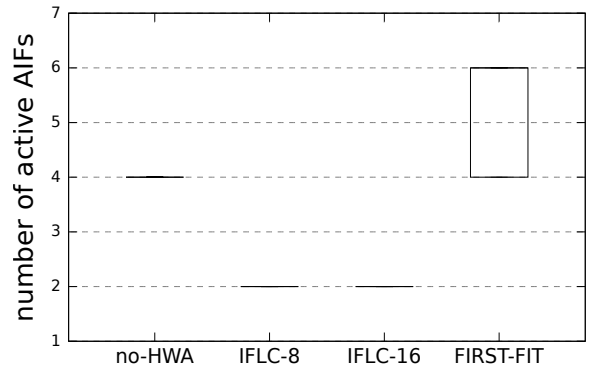
- The number of active AIFs reduces with the number of HWA. More precisely, for no-HWA feasible solutions refer to instances with less stringent time constraints. The corresponding number of active AIFs is equal to 4, that is, 4 AIFs are deployed to respect the target time. This number decreases by up to 50% thanks to HWA: both IFLC-8 and IFLC-16 yield a lower number of active AIFs (i.e, 2 AIFs) when the same time constraints apply.
- For FIRST-FIT, the minimum number of active AIFs is higher with core-edge setting, as it is more flexible than edge-edge in placing edge AIFs. In that case, if the stopping point is not yet achieved (i.e., possibility of decreasing the local training time), FIRST-FIT will keep increasing the number of AIFs. This confirms the previous results showing that the local training time is lower with core-edge.



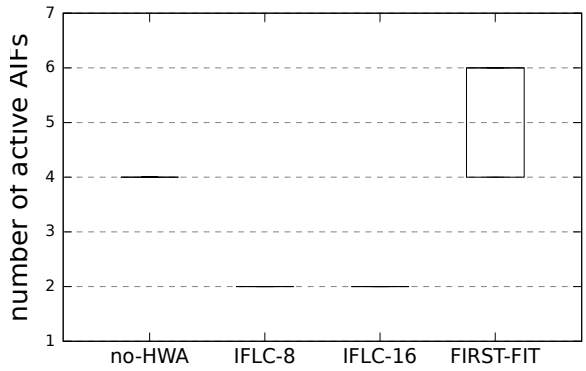
(a) case D-S: edge-edge



(b) case S-S: edge-edge



(c) case D-S: core-edge



(d) case S-S: core-edge

Figure 6.8: Distribution of the number of active AIFs.

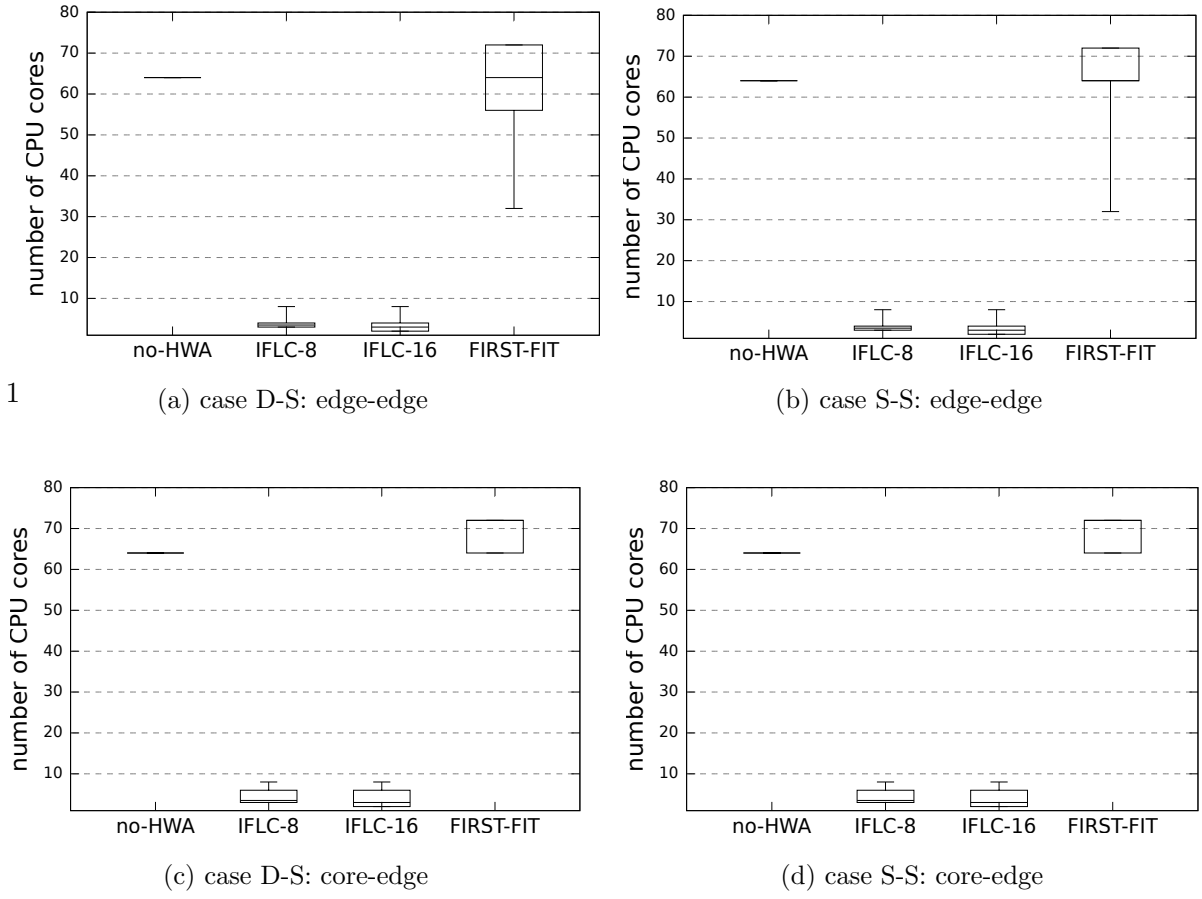
- IFLC-8 yields the same number of active AIFs as IFLC-16. This can be explained by the fact that IFLC promotes using HWA instead of increasing the number of active AIFs to reduce the local training time. As HWA may not be available on some physical nodes with IFLC-8, the latter chooses the same number of active AIFs as IFLC-16 but with higher cpu resources.

- higher CPU resources are needed to reduce the local training time and consequently the E2E training time for FIRST-FIT and no-HWA, which is correlated with the higher number of active AIFs even with less strict time constraints. Also, FIRST-FIT achieves the same minimum cost as IFLC for a small number of instances with edge-edge placement:

  for theses instances, the stopping point is achieved with a low number of active AIFs w.r.t the other instances, which results in a lower number of CPU cores.

- The distribution of CPU cores is slightly different when comparing IFLC-8 and IFLC-16. As already explained, when stringent time constraints apply, IFLC-8 yield the same number of active AIFs as IFLC-16. However, the former may not have available HWA on nodes with low cpu resources which leads to solutions with slightly higher processing capacities.

Overall, the advantage of an IFLC scheme is the capability of exploiting HWA, leading to the lowest computational costs, as less CPU resources are needed.



1

(a) case D-S: edge-edge

(b) case S-S: edge-edge

(c) case D-S: core-edge

(d) case S-S: core-edge

Figure 6.9: Distribution of the total number of CPU cores.

6.7 Conclusion

In this chapter, we proposed a federated learning system control scheme to permit dynamic selection of hardware accelerators in artificial intelligence function orchestration for in-network applications. Our scheme is designed to decrease the number of learning stragglers, while making efficient use of heterogeneous computing resources.

A major highlight is that we demonstrated how adaptive hardware acceleration can reduce the number of stragglers up to 100%, when comparing to first-fit scheduling of federated learning clients across a distributed network, and the cases where hardware accelerators are not used by these clients. We also avoid with our schemes a random or systemic use of hardware acceleration, which may lead to either too high variance in the end-to-end training times on the one hand, or avoidable computational overhead on the other hand.

Our contribution also originally identifies the necessity to combine network delays with distributed training delays when seeking efficient learning solutions. In our analysis, we also cope with stochastic variations in both network delays and local training times when designing our in-network federated learning control scheme.

Chapter 7

Conclusion

Contents

7.1 Conclusion

Over the past decade, the number of mobile users has increased exponentially with the proliferation of 5G multimedia applications that are targeting very low-latency and high reliability. This increasing demands for data services led to new challenges related to end-to-end latency, computing capabilities, energy consumption and the number of connected devices. These challenges do not only affect network operators but also service and application providers. Recently, a distributed network design known as Multi-access Edge Computing was introduced as one of the 5G key enabler to overcome the above challenges. In practice, MEC supports low latency as well as intensive computation by running applications closer to end users. However, several challenges related to resource scheduling can be encountered in MEC systems due to limited capacities of the physical infrastructure. Throughout this thesis, we tackled specific challenges in MEC environments that are related to resource orchestration. Based on the state-of-the art analysis, the central questions for this research addressed the optimization of resource orchestration in MEC where we considered the minimization of the latency budget to support QoS requirements in different settings.

- In Chapter 3, we considered the radio access disaggregation where the centralized functions that can be shared among several base stations are located at the MEC host. In order to optimize the assignment decision of base stations to a set of MEC hosts. We proposed a data-driven approach based on a spatial clustering model; formulated using Integer Linear Programming; that groups together base stations based on their spatio-temporal behavior. Afterwards, we applied an orchestration model on the resulting clusters to assign them to the available MEC hosts. The main goal is to minimize the deployment and latency costs. Indeed, we noticed a trade-off relationship between the framework complexity in terms of time and space (i.e., execution time and memory consumption), and its performance in terms of latency cost.

- We then proposed in Chapter 4 an enhancement of the clustering model where we suggested a collection of pairwise clustering models that group together base stations depending on their traffic demands variation. The main goal of this framework is not only to reduce the temporal and spatial complexity but also to provide a higher control over MEC hosts capacity. The most robust clustering models reduced the violation rate by up to 62% when compared to a baseline solution. Also, the proposed solutions helps reduce both memory usage and execution time, by

46% and 50%, respectively, in comparison to this baseline solution.

- With the integration of AI and edge networks towards fulfilling network management tasks in real-time, and given the high impact of MEC paradigm on meeting the targeted time requirements, we addressed the resource orchestration problem from an application point of view. More precisely, we considered a FL-based anomaly detection from the state-of-the-art which consists of deploying a variable number of FL participants (i.e., FL clients). This study comes in line with the objective of network reconfiguration automation using AIML techniques in providing efficient resource management when deploying AIML functions by leveraging MEC paradigm to ensure low communication delays. Thus, in Chapter 5, we considered the problem of finding the optimal placement for a set of artificial intelligence functions running federated learning at the edge. The goal is to respect the targeted end-to-end learning time while considering the possibility of using hardware acceleration to reduce the local learning time. We mathematically formulated the placement problem using Mixed-Integer Linear Programming where we considered different time components such as propagation delays and local training times of the AIF, the placement of the FL server, the FL client AIFs selection and the local training time efficiency using hardware acceleration.

- As the local training time highly depends on the available computing resources and the amount of the data to train, and given that the training time highly contributes to the end-to-end training time, in Chapter 6 we studied and evaluated the impact of system heterogeneity on end-to-end learning time. Hence, we mathematically formulated the mentioned problem as a MILP while increasing the number of FL participants that positively contribute to the FL learning (i.e., respect the targeted delays). By extensive simulation we showed how our approach outperforms a first-fit algorithm where we increased by up to 100% the number of FL clients contributing to the FL task.

7.2 Future Research Directions

In this thesis, we tackled a set of resource orchestration challenges in MEC. In the following, we present other research questions that can be tackled in this area.

7.2.1 Online Resource Orchestration in MEC

Based on the BS-to-MEC orchestration framework presented in Chapters 3 and 4, future works may further push time-execution requirements barrier for real-time MEC orchestration, integrating real-time traffic prediction. To do so, one may consider the dynamic arrival of tasks at MEC hosts where the assignment decision is done based on a previous short amount of time. Reinforcement learning could also be used in that case to improve the assignment decision over time.

Another research direction in this area can be the usage for combined MEC and vRAN orchestration, including multiple decision points such as in functional splitting, also addressing different objectives such as based on additional quality-of-service criteria.

7.2.2 Management of Stragglers in MEC-FL environments

One of the shortcomings of the AIF placement framework we proposed in Chapter 6 is the non consideration of the data arrival time from nodes where data is collected (i.e., data sources, can be a mobile application, for example), to the edge servers where the AIML models are trained. Hence, considering the challenge of interfacing the AIFs with a data pipeline system is an interesting research direction.

Another future work could cover the refining of the aggregation functions at the federated learning server level, in order to further increase the quality of the distributed learning. Moreover, we plan to work on scaling the resulting learning systems by means of split learning to cover multiple heterogeneous learning domains.

7.2.3 Service Differentiation and Network Reliability Support

Despite the fact that MEC servers can be densely distributed at the edge, resource limitation remains one of the biggest challenges. As for the telecoms field, NFV and SDN paradigms have been proposed to efficiently manage the available resources, pushing back the barriers to the design of virtualized infrastructures on both access and core networks, with a decoupling between the data plane and the control plane. This led to the emergence of the Network Slicing concept, initially designed to meet the needs of highly specific, heterogeneous applications such as uRLLC, eMBB and mMTC from 5G services. An interesting direction is to extend the framework in Chapter 4 to consider

the placement of both radio access and core networks functions taking into account heterogeneous requirements for resource orchestration while prioritizing services with stringent requirements.

Reliability is another important challenge to tackle to ensure the requirements of 5G services. Given the *Network Function Set* concept from ETSI technical specification [87], an interesting research direction is to study the reliability guarantee by duplicating the network functions.

Chapter 8

Résumé

Contents

8.1 Paradigme du Multi-access Edge Computing

Le paradigme MEC a été initialement développé pour exécuter des services informatiques près des terminaux afin de réduire la latence et d'améliorer l'expérience utilisateur. Malgré le fait que les hôtes d'infrastructure MEC peuvent être densément distribués à la périphérie, la limitation des ressources et la robustesse contre les fluctuations du trafic sont des défis importants à relever par les fournisseurs de services réseau. De nouvelles technologies telles que NFV et SDN ont été proposées au cours de la dernière décennie; leur considération dans la conception de l'architecture de réseau est de pousser les barrières technologiques vers les infrastructures virtualisées aux sous-systèmes RAN [1], incluant à la fois la centralisation du contrôle et la virtualisation et la softwarisation de toutes les fonctions réseau impliquées.

La virtualisation des fonctions radio conduit à une flexibilité supplémentaire dans un segment historiquement plus rigide que les réseaux centraux, en raison de la moindre importance du routage dans ces environnements. Cette flexibilité peut aider à répondre aux demandes croissantes et imprévisibles des utilisateurs mobiles, et permet également l'utilisation de matériel standard pour réduire les coûts pour MNO et retarder les dépenses en capital. En outre, la technologie MEC permet de faire face à la variation de la demande des utilisateurs, car la reconfiguration du réseau devient une opération plus facile à effectuer [2]. En effet, alors que les infrastructures MEC sont reconnues comme un catalyseur clé de la 5G, l'inverse est également vrai : la 5G peut être considérée comme un catalyseur clé pour les infrastructures MEC, grâce à la technologie NFV [3]. Le déploiement d'installations de virtualisation dans le réseau d'accès, pour les fonctions 5G et RAN, peut donc favoriser le déploiement d'éléments d'infrastructure MEC. Le déploiement des serveurs d'applications à proximité des utilisateurs finaux peut augmenter les débits binaires des utilisateurs et réduire la latence de bout en bout [4]. Notez que MEC a besoin d'une plateforme de virtualisation pour déployer ses applications en périphérie. Dans ce cas, la plateforme NFV peut être utilisée pour déployer des applications VNFs et MEC.

L'emplacement des hôtes MEC est actuellement envisagé par les opérateurs de télécommunications pour se produire à CO et/ou PoP. La distribution des hôtes MEC horizontalement sur différents segments de réseau d'accès et verticalement sur différentes couches du réseau de retour est nécessaire pour répondre aux exigences de latence d'accès et de fiabilité. En règle générale, les hôtes MEC sont censés être situés entre les stations de base et le réseau central [5]. Une représentation de référence de

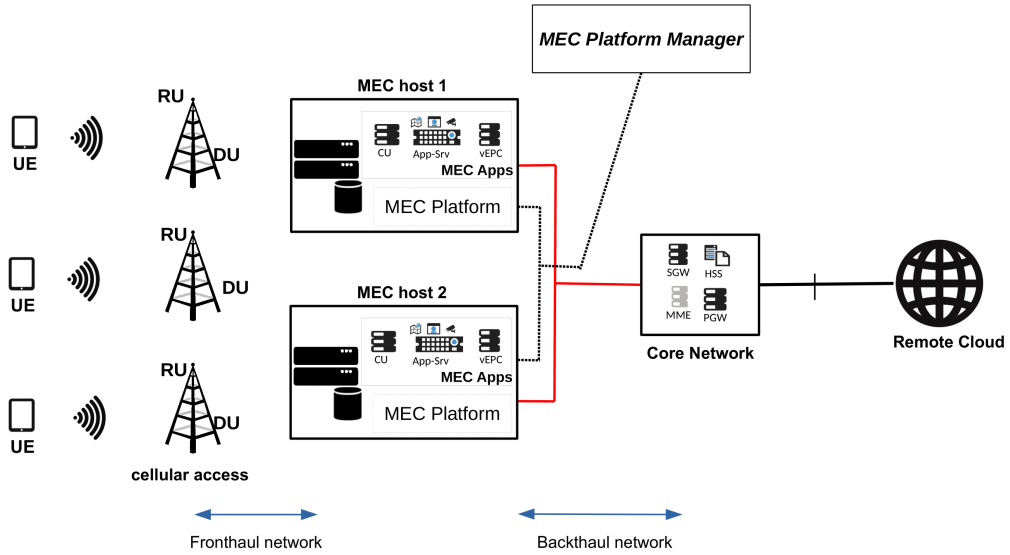


Figure 8.1: Représentation de l'infrastructure MEC de référence.

l'infrastructure MEC se trouve dans la Figure 8.1. À strictement parler [6], un serveur MEC (cloudlet ou MEC facility) fait référence aux serveurs matériels appartenant à l'infrastructure de virtualisation ; il peut être générique ou basé sur NFVI, et dans ce cas l'hôte MEC peut être déployé comme un VNF, prenant éventuellement en charge le découpage du réseau [3]. La plateforme MEC rq est responsable de la gestion des applications MEC.

Lorsque différents hôtes MEC sont déployés sur le réseau d'accès d'un opérateur, il forme un nuage distribué appelé 'MEC system '.

Afin d'avoir un contrôle complet du déploiement des services sur l'infrastructure MEC, les normes ETSI nécessitent le développement d'éléments de service d'orchestration, dans le but de gérer efficacement les ressources disponibles sur les hôtes MEC. Par conséquent, l'automatisation de la tâche susmentionnée est considérée comme l'un des défis importants à relever. En outre, étant donné que les utilisateurs finaux disposent de la fonctionnalité mobile au sein du réseau mobile, le soutien à la mobilité est une autre exigence de l'ETSI-MEC pour assurer la continuité des services. Les demandes

MEC peuvent ensuite être divisées en demandes dépendant de l'État et indépendantes de l'État. Le premier est spécifique à l'utilisateur où une partie ou la totalité des informations doivent être conservées alors que le second n'est pas lié à l'activité de l'utilisateur.

En outre, trois grandes catégories de cas d'utilisation de MEC ont été identifiées par ETSI [7]. Les applications orientées vers le consommateur, elles sont directement liées à l'utilisateur final (équipement utilisateur) tels que les jeux et la réalité augmentée. Les applications axées sur l'opérateur, qui sont des services qui ne sont peut-être pas directement liés aux utilisateurs, mais à des tiers, comme les applications de sécurité et de sûreté, le suivi des dispositifs terminaux, etc. Enfin, le QoE applications d'amélioration qui visent à améliorer l'expérience utilisateur tout en se concentrant sur l'optimisation du réseau. Par exemple, on peut citer la mise en cache de contenu, l'optimisation du déploiement des hôtes MEC et la planification des ressources. Il convient de mentionner que cette thèse se concentre sur la troisième catégorie de cas d'utilisation. Dans ce qui suit, nous présentons les différents défis rencontrés dans les environnements MEC.

8.2 Énoncé du Problème et Défis

Les applications MEC ont généralement un ensemble d'exigences telles que la capacité de calcul, l'efficacité énergétique et la latence. Comme les systèmes MEC sont déployés sur des serveurs dont les capacités sont limitées (c.-à-d. capacité de stockage et de traitement), la disponibilité des ressources peut changer au fil du temps, ce qui nécessite la migration des applications MEC d'un hôte MEC à un autre. Notez que chaque emplacement peut avoir un coût différent en termes de performance, de déploiement, ou les deux. Ainsi, le déploiement d'une application MEC au meilleur emplacement (c.-à-d., l'hôte MEC le plus proche avec suffisamment de ressources) peut ne pas toujours être le meilleur choix. À cette fin, les systèmes MEC devraient pouvoir décider du placement des demandes MEC tout en tenant compte du fait que la décision de placement peut changer au fil du temps à mesure que les conditions évoluent.

Dans cette thèse, nous considérons les questions liées à la planification des ressources dans MEC. En général, la planification des ressources fait référence aux techniques utilisées pour affecter un ensemble de ressources disponibles aux utilisateurs mobiles afin d'accomplir une tâche spécifique à un moment donné. La conception de stratégies efficaces de planification des ressources comporte deux

volets : (i) atteindre la qualité de service souhaitée au niveau de l'utilisateur et (ii) optimiser les coûts au niveau du fournisseur de services de périphérie. Les problèmes de planification des ressources dans les environnements périphériques font l'objet d'un grand intérêt dans la littérature. Par exemple, [8] discute des travaux de recherche qui sont liés à la planification des ressources en informatique de pointe. Il classe les actions possibles dans la planification des ressources en trois catégories : (i) le déchargement de calcul [9], (ii) l'orchestration des ressources [10] et (iii) le provisionnement des ressources [11]. Le délestage de calcul est une solution prometteuse pour libérer UE des tâches de calcul intensives. La décision de déchargement est prise en fonction de plusieurs exigences telles que la latence, le coût et la consommation d'énergie. Comme présenté par [8], le délestage de calcul peut être classé en fonction de la direction de délestage c.-à-d., UE-à-bord, bord-à-nuage, bord-à-bord, ... etc., ou en fonction de la granularité, c.-à-d., des tâches partiellement ou totalement déchargées.

Un autre problème de recherche dans l'environnement MEC est l'orchestration des ressources. Il consiste à allouer de manière flexible des ressources de calcul, de communication ou de stockage afin de garantir un QoS donné et peut également considérer conjointement plusieurs ressources. Notez que dans cette thèse, l'orchestration des ressources fait référence à l'allocation des ressources, au placement des services et des applications ou aux deux. Enfin, le provisionnement des ressources consiste à allouer la quantité appropriée de ressources pour garantir les exigences de QoS. Notez que la différence entre l'orchestration des ressources et le provisionnement des ressources est que le premier attribue les ressources disponibles aux utilisateurs pour assurer un service donné tandis que le second assure la disponibilité des ressources en cas de besoin.

8.3 Questions de Recherche et Contributions

L'émergence de nouvelles applications omniprésentes avec des exigences strictes et hétérogènes en termes de latence et de bande passante a conduit à l'apparition du paradigme MEC pour faire face à la limitation du cloud computing traditionnel. D'autre part, des paradigmes de virtualisation tels que NFV et SDN ont été initialement proposés pour gérer efficacement les ressources disponibles, ce qui a conduit à l'apparition d'un concept de découpage de réseau pour répondre aux exigences spécifiques et hétérogènes des applications et des services. Par conséquent, la conception de tranches adaptées aux applications, les algorithmes efficaces d'allocation des ressources et la gestion résiliente des réseaux et des systèmes sont désormais considérés comme les nouveaux défis de recherche à relever dans le

domaine des réseaux et des télécommunications.

L'objectif de cette thèse est d'automatiser et d'optimiser l'allocation des ressources dans un environnement 5G-MEC en tenant compte des enjeux d'évolutivité. Les contributions de cette thèse visent à répondre aux questions de recherche suivantes :

1. *D'un point de vue de l'infrastructure MEC* : dans le RAN désagrégé où les fonctions de traitement radio sont divisées en CU et DU, comment décider efficacement quel groupe de BSs doit être desservi par une UC donnée en fonction de la capacité des hôtes MEC, la latence d'accès et les coûts de déploiement ? En outre, comment MNOs pourrait-il utiliser l'analyse des données pour optimiser la solution d'orchestration en la rendant évolutive et robuste contre un déploiement en temps quasi réel ? Ces deux questions sont abordées dans les chapitres 3 et 4.

2. textitD'un point de vue applicatif MEC : lors du déploiement de modèles IA/ML (Intelligence Artificielle/Machine Learning) distribués à la périphérie, comment contrôler la variation des temps d'apprentissage de bout en bout en raison de l'hétérogénéité des ressources sur les hôtes MEC ? Cette question de recherche est détaillée dans les chapitres 5 et 6.

Cette thèse vise à fournir une compréhension approfondie des défis d'orchestration des ressources dans les environnements MEC tout en proposant plusieurs approches novatrices qui les abordent efficacement à différents niveaux (c.-à-d., niveau de l'infrastructure et niveau de l'application). Les principales contributions sont présentées ci-dessous.

8.3.1 Niveau d'infrastructure MEC

8.3.1.1 Clustering spatial pour l'affectation de stations de base aux hôtes MEC

Dans ce travail, nous nous concentrons sur l'optimisation des tâches d'orchestration MEC pour faire face aux nouvelles applications 5G omniprésentes où nous considérons la complexité et l'évolutivité des stations de base par rapport aux serveurs MEC problème d'affectation. Nous relevons ce défi pour inclure des objectifs secondaires aux algorithmes existants pour l'orchestration MEC, et en particulier pour le problème de trouver des assignations de stations de base aux hôtes MEC. Le cadre proposé comporte deux étapes. Nous appliquons d'abord un modèle de clustering spatial sur l'ensemble des BSs à la phase de prétraitement puis nous résolvons le problème d'affectation des clusters de BSs

résultants aux hôtes MEC disponibles en adaptant le modèle d'orchestration. Notez que l'opération d'affectation a un coût défini par la latence d'accès. Nous considérons également la réaffectation des ressources lorsqu'un VM desservant un utilisateur ou un ensemble d'utilisateurs est migré d'un hôte MEC à un autre.

Le modèle de regroupement spatial regroupe les SR en fonction de leur profil des demandes de trafic (c.-à-d. le comportement spatio-temporel) afin de minimiser la variance des demandes de trafic au sein de chaque groupe de SR. L'objectif principal de cette proposition est de réduire la complexité spatiale et temporelle (c'est-à-dire le temps d'exécution et la consommation de mémoire). Les résultats obtenus à partir de la simulation étendue contre les demandes réelles de trafic montrent comment notre proposition réduit la complexité du temps et de l'espace compte tenu de l'algorithme de base. Même si la technique proposée engendre des coûts supplémentaires, sa robustesse permet d'exécuter le framework en temps réel. D'autres résultats sont disponibles dans le chapitre 3.

8.3.1.2 Modèles de clustering robustes pour l'affectation des stations de base aux hôtes MEC

Alors que le modèle de clustering BS présenté précédemment augmente les coûts des utilisateurs en termes de latence et puisque les services 5G visent à assurer utlra-Nous proposons d'étendre le travail précédent pour améliorer la décision d'orchestration en termes de latence et le degré de contrôle de la capacité des hôtes MEC en réduisant la violation de capacité qui peut se produire en raison de la fluctuation du trafic. Pour ce faire, nous proposons de regrouper les BSs par paires en fonction d'un ensemble de critères. Le travail correspondant est détaillé dans le chapitre 4. Les modèles proposés sont évalués à l'aide d'un ensemble de données réelles. Notez que le cadre d'orchestration est évalué dans un paramètre hors ligne sous différents paramètres tout en variant le nombre et la capacité des hôtes MEC disponibles.

8.3.2 Niveau d'application MEC

8.3.2.1 Placement optimal des applications MEC pour un cadre d'apprentissage fédéré

Comme le paradigme MEC permet d'apporter des ressources pour l'informatique AIML au réseau périphérique où les données à traiter sont situées, nous proposons d'étendre le travail susmentionné pour optimiser les décisions d'allocation des ressources au niveau de l'application MEC. Nous considérons que le modèle AIML est déployé comme une application MEC. Ainsi, nous abordons le problème

de placer AIFs exécutant l'apprentissage fédéré contre les données de surveillance d'infrastructure de réseau de calcul connecté, pour les environnements où l'introduction de l'edge computing vient avec un ensemble hétérogène et important d'éléments de calcul et de mise en réseau, nécessitant de faibles performances de latence. En particulier, nous utilisons comme cas d'utilisation de référence le AIF de détection d'anomalie FL proposé dans [12], adapté pour l'infrastructure 5G. Ce cadre d'apprentissage fédéré utilise un FIA de serveur d'apprentissage fédéré et un nombre variable de FIA de périphérie: la tâche d'apprentissage est distribuée aux FIA de périphérie par des données de surveillance d'équilibrage de charge entre eux, où les FIA de périphérie interagissent via le serveur pour les mises à jour du modèle d'apprentissage.

L'objectif visé est de réduire le temps d'apprentissage de bout en bout afin de respecter le seuil de temps imposé par la spécification de l'application. Pour ce faire, nous nous concentrons sur le placement des AIF utilisant HWA. Nous modélisons le comportement de l'apprentissage fédéré et du point d'inférence associé pour guider la décision de placement, en tenant compte de la contrainte spécifique et du comportement empirique d'un cas d'utilisation de détection d'anomalies d'infrastructure virtualisée. En plus de l'accélération matérielle, nous considérons la tendance spécifique du temps d'entraînement lors de la distribution de la formation sur un réseau, en utilisant des distributions linéaires empiriques à la pièce et nous modélisons le problème de placement comme un MILP. Les résultats de simulation montrent l'impact que l'accélération matérielle peut avoir dans la décision du nombre de AIF déployés, tout en divisant par un facteur pertinent le temps de formation distribué. Plus de détails sur cette contribution sont disponibles dans le chapitre 5.

8.3.2.2 Placement d'applications MEC pour contrôler les retardataires dans un environnement d'apprentissage fédéré

Nous étendons le modèle de placement susmentionné où nous introduisons le IFLC, qui est un schéma adaptatif pour l'utilisation des HWAs dans les systèmes distribués pour compenser les variations de bout en bout du réseau et des retards d'apprentissage conduisant à des retardateurs. Nous allons au-delà des travaux existants, en reformulant le modèle pour contrôler les retardateurs, en définissant une modélisation affinée de la latence d'entraînement de bout en bout, et en proposant un algorithme de résolution optimale polynomiale supportant ainsi l'orchestration en temps quasi réel des FL-AIF. Grâce à une évaluation approfondie des performances, nous mettons en évidence l'impact

de l'utilisation d'accélérateurs matériels pour atteindre un ratio plus élevé de participants LF qui contribuent positivement à l'effort d'apprentissage, tandis que ce ratio est augmenté jusqu'à 100% par rapport à un algorithme de premier ajustement. Plus de détails sont disponibles dans le chapitre 6.

Bibliography

[1] A. Garcia-Saavedra, G. Iosifidis, X. Costa-Perez, and D. J. Leith, "Joint optimization of edge computing architectures and radio access networks," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 11, pp. 2433–2443, 2018.

[2] J. Zhang, D. Zeng, L. Gu, H. Yao, and M. Xiong, "Joint optimization of virtual function migration and rule update in software defined nfv networks," in *GLOBECOM 2017-2017 IEEE Global Communications Conference*. IEEE, 2017, pp. 1–5.

[3] M. E. ISG, "'multi-access edge computing (mec); support for network slicing," *ETSI, Sophia-Antipolis, France, Tech. Rep. GR MEC*, vol. 24, 2019.

[4] M. ETSI, "Multi-access edge computing (mec); radio network information api," *ETSI GS MEC*, vol. 12, p. V2, 2019.

[5] S. Kekki, W. Featherstone, Y. Fang, P. Kuure, A. Li, A. Ranjan, D. Purkayastha, F. Jiangping, D. Frydman, G. Verin *et al.*, "Mec in 5g networks," *ETSI white paper*, vol. 28, no. 2018, pp. 1–28, 2018.

[6] M. E. Computing, "Deployment of mobile edge computing in an nfv environment," *ETSI Group Report MEC*, vol. 17, p. V1, 2018.

[7] ETSI, "Multi-access edge computing (mec); use cases and requirements," *ETSI GS MEC*, 2023.

[8] Q. Luo, S. Hu, C. Li, G. Li, and W. Shi, "Resource scheduling in edge computing: A survey," *IEEE Communications Surveys Tutorials*, vol. 23, no. 4, pp. 2131–2165, 2021.

[9] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE communications surveys & tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[10] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1657–1681, 2017.

[11] T. L. Duc, R. G. Leiva, P. Casari, and P.-O. Östberg, "Machine learning methods for reliable resource provisioning in edge-cloud computing: A survey," *ACM Computing Surveys (CSUR)*, vol. 52, no. 5, pp. 1–39, 2019.

[12] S. B. Ruba, N. E.-H. Yellas, and S. Secci, "Anomaly detection for 5g softwarized infrastructures with federated learning," in *2022 1st International Conference on 6G Networking (6GNet)*, 2022, pp. 1–4.

[13] A. Reznik, L. M. C. Murillo, F. Fontes, C. Turyagyenda, C. Wehner, and Z. Zheng, "Cloud ran and mec: A perfect pairing," 2018.

[14] A. Filali, A. Abouaomar, S. Cherkaoui, A. Kobbane, and M. Guizani, "Multi-access edge computing: A survey," *IEEE Access*, vol. 8, pp. 197 017–197 046, 2020.

[15] L. Yala, P. A. Frangoudis, and A. Ksentini, "Latency and availability driven vnf placement in a mec-nfv environment," in *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2018, pp. 1–7.

[16] R. Cziva, C. Anagnostopoulos, and D. P. Pezaros, "Dynamic, latency-optimal vnf placement at the network edge," in *Ieee infocom 2018-ieee conference on computer communications*. IEEE, 2018, pp. 693–701.

[17] Y. Nam, S. Song, and J.-M. Chung, "Clustered nfv service chaining optimization in mobile edge clouds," *IEEE Communications Letters*, vol. 21, no. 2, pp. 350–353, 2016.

[18] A. Ceselli, M. Premoli, and S. Secci, "Mobile edge cloud network design optimization," *IEEE/ACM Transactions on Networking*, vol. 25, no. 3, pp. 1818–1831, 2017.

[19] H. Yu, F. Musumeci, J. Zhang, Y. Xiao, M. Tornatore, and Y. Ji, "Du/cu placement for c-ran over optical metro-aggregation networks," in *Optical Network Design and Modeling: 23rd IFIP WG 6.10 International Conference, ONDM 2019, Athens, Greece, May 13–16, 2019, Proceedings 23*. Springer, 2020, pp. 82–93.

[20] I. GSTR-TN5G, "Transport network support of imt-2020/5g," *Geneva, Switzerland, Oct*, 2018.

[21] W. da Silva Coelho, A. Benhamiche, N. Perrot, and S. Secci, "On the impact of novel function mappings, sharing policies, and split settings in network slice design," in *2020 16th International Conference on Network and Service Management (CNSM)*, 2020, pp. 1–9.

[22] N. Makris, V. Passas, T. Korakis, and L. Tassiulas, "Employing mec in the cloud-ran: An experimental analysis," in *Proceedings of the 2018 on Technologies for the Wireless Edge Workshop*, 2018, pp. 15–19.

[23] W. Diego, "Evolution toward the next generation radio access network," in *2020 IFIP Networking Conference (Networking)*, 2020, pp. 685–685.

[24] S. K. Singh, R. Singh, and B. Kumbhani, "The evolution of radio access network towards open-ran: Challenges and opportunities," in *2020 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, 2020, pp. 1–6.

[25] B. Wu, J. Zeng, L. Ge, Y. Tang, and X. Su, "A game-theoretical approach for energy-efficient resource allocation in mec network," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, 2019, pp. 1–6.

[26] M. Zheng, Y. Zhao, X. Zhang, C.-Z. Xu, and X. Li, "A feedback prediction model for resource usage and offloading time in edge computing," in *International Conference on Cloud Computing*. Springer, 2018, pp. 235–247.

[27] I. Alghamdi, C. Anagnostopoulos, and D. P. Pezaros, "Time-optimized task offloading decision making in mobile edge computing," in *2019 Wireless Days (WD)*, 2019, pp. 1–8.

[28] J. Li, H. Gao, T. Lv, and Y. Lu, "Deep reinforcement learning based computation offloading and resource allocation for mec," in *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, 2018, pp. 1–6.

[29] X. Xu, X. Zhang, X. Liu, J. Jiang, L. Qi, and M. Z. A. Bhuiyan, "Adaptive computation offloading with edge for 5g-envisioned internet of connected vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 8, pp. 5213–5222, 2021.

[30] L. Chen, D. Yang, D. Zhang, C. Wang, J. Li *et al.*, "Deep mobile traffic forecast and complementary base station clustering for c-ran optimization," *Journal of Network and Computer Applications*, vol. 121, pp. 59–69, 2018.

[31] L. Chen, L. Liu, X. Fan, J. Li, C. Wang, G. Pan, J. Jakubowicz, and T.-M.-T. Nguyen, "Complementary base station clustering for cost-effective and energy-efficient cloud-ran," in *2017 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computed, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, 2017, pp. 1–7.

[32] S. Ntalampiras and M. Fiore, "Forecasting mobile service demands for anticipatory mec," in *2018 IEEE 19th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, 2018, pp. 14–19.

[33] M. Bouet and V. Conan, "Mobile edge computing resources optimization: A geo-clustering approach," *IEEE Transactions on Network and Service Management*, vol. 15, no. 2, pp. 787–796, 2018.

[34] A. Ceselli, M. Fiore, A. Furno, M. Premoli, S. Secci, and R. Stanica, "Prescriptive analytics for mec orchestration," in *2018 IFIP Networking Conference (IFIP Networking) and Workshops*. IEEE, 2018, pp. 1–9.

[35] A. Furno, D. Naboulsi, R. Stanica, and M. Fiore, "Mobile demand profiling for cellular cognitive networking," *IEEE Transactions on Mobile Computing*, vol. 16, no. 3, pp. 772–786, 2017.

[36] H. Djeddal, L. Touzari, A. Giovanidis, C.-D. Phung, and S. Secci, "Hyperbolic k-means for traffic-aware clustering in cloud and virtualized rans," *Computer Communications*, vol. 176, pp. 258–271, 2021.

[37] D. Naboulsi, A. Mermouri, R. Stanica, H. Rivano, and M. Fiore, "On user mobility in dynamic cloud radio access networks," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, 2018, pp. 1583–1591.

[38] R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, and O. M. Caicedo, "A comprehensive survey on machine learning for networking: evolution, applications

and research opportunities," *Journal of Internet Services and Applications*, vol. 9, no. 1, pp. 1–99, 2018.

[39] J. Bendriss, I. G. Ben Yahia, P. Chemouil, and D. Zeghlache, "Ai for sla management in programmable networks," in *DRCN 2017 - Design of Reliable Communication Networks; 13th International Conference*, 2017, pp. 1–8.

[40] V. C. Emeakaroha, I. Brandic, M. Maurer, and S. Dustdar, "Low level metrics to high level slas - lom2his framework: Bridging the gap between monitored metrics and sla parameters in cloud environments," in *2010 International Conference on High Performance Computing Simulation*, 2010, pp. 48–54.

[41] A. Diamanti, J. M. S. Vílchez, and S. Secci, "An ai-empowered framework for cross-layer softwarized infrastructure state assessment," *IEEE Transactions on Network and Service Management*, vol. 19, no. 4, pp. 4434–4448, 2022.

[42] S. Deng, H. Zhao, W. Fang, J. Yin, S. Dustdar, and A. Y. Zomaya, "Edge intelligence: The confluence of edge computing and artificial intelligence," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7457–7469, 2020.

[43] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1738–1762, 2019.

[44] *Architecture enhancements for 5G System to support network data analytics services*, 3GPP TS 23.288, Rev. 17.1.0, June 2021, Accessed: August 10, 2023. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3579

[45] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[46] Y. Jeon, H. Jeong, S. Seo, T. Kim, H. Ko, and S. Pack, "A distributed nwdaf architecture for federated learning in 5g," in *2022 IEEE International Conference on Consumer Electronics (ICCE)*, 2022, pp. 1–2.

[47] P. Rajabzadeh and A. Outtagarts, "Federated learning for distributed nwdaf architecture," in *2023 26th Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, 2023, pp. 24–26.

[48] A. Sacco, F. Esposito, and G. Marchetto, "A federated learning approach to routing in challenged sdn-enabled edge networks," in *2020 6th IEEE Conference on Network Softwarization (NetSoft)*, 2020.

[49] S. Garg, T. Bag, and A. Mitschele-Thiel, "Decentralized machine learning based network data analytics for cognitive management of mobile communication networks," in *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, 2022, pp. 1–9.

[50] R. Schlegel, S. Kumar, E. Rosnes, and A. G. i Amat, "Straggler-resilient secure aggregation for federated learning," in *2022 30th European Signal Processing Conference (EUSIPCO)*, 2022, pp. 712–716.

[51] W. Y. B. Lim, J. S. Ng, Z. Xiong, J. Jin, Y. Zhang, D. Niyato, C. Leung, and C. Miao, "Decentralized edge intelligence: A dynamic resource allocation framework for hierarchical federated learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 3, pp. 536–550, 2022.

[52] W. Y. B. Lim, J. S. Ng, Z. Xiong, D. Niyato, C. Miao, and D. I. Kim, "Dynamic edge association and resource allocation in self-organizing hierarchical federated learning networks," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 12, pp. 3640–3653, 2021.

[53] H. Ko, J. Lee, S. Seo, S. Pack, and V. C. M. Leung, "Joint client selection and bandwidth allocation algorithm for federated learning," *IEEE Transactions on Mobile Computing*, vol. 22, no. 6, pp. 3380–3390, 2023.

[54] L. Yu, R. Albelaihi, X. Sun, N. Ansari, and M. Devetsikiotis, "Jointly optimizing client selection and resource management in wireless federated learning for internet of things," *IEEE Internet of Things Journal*, vol. 9, no. 6, pp. 4385–4395, 2022.

[55] L. U. Khan, S. R. Pandey, N. H. Tran, W. Saad, Z. Han, M. N. H. Nguyen, and C. S. Hong, "Federated learning for edge networks: Resource optimization and incentive mechanism," *IEEE Communications Magazine*, vol. 58, no. 10, pp. 88–93, 2020.

[56] R. Zeng, S. Zhang, J. Wang, and X. Chu, "Fmore: An incentive scheme of multi-dimensional auction for federated learning in mec," in *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*, 2020, pp. 278–288.

[57] Y. Cui, K. Cao, G. Cao, M. Qiu, and T. Wei, "Client scheduling and resource management for efficient training in heterogeneous iot-edge federated learning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 8, pp. 2407–2420, 2022.

[58] J. Lee, J. K. Eshraghian, K. Cho, and K. Eshraghian, "Adaptive precision cnn accelerator using radix-x parallel connected memristor crossbars," *arXiv preprint arXiv:1906.09395*, 2019.

[59] H. Giefers, P. Staar, C. Bekas, and C. Hagleitner, "Analyzing the energy-efficiency of sparse matrix multiplication on heterogeneous systems: A comparative study of gpu, xeon phi and fpga," in *2016 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2016, pp. 46–56.

[60] B. Betkaoui, D. B. Thomas, and W. Luk, "Comparing performance and energy efficiency of fpgas and gpus for high productivity computing," in *2010 International Conference on Field-Programmable Technology*, 2010, pp. 94–101.

[61] M. Qasaimeh, K. Denolf, J. Lo, K. Vissers, J. Zambreno, and P. H. Jones, "Comparing energy efficiency of cpu, gpu and fpga implementations for vision kernels," in *2019 IEEE International Conference on Embedded Software and Systems (ICESS)*, 2019, pp. 1–8.

[62] J. Xu, S.-L. Huang, L. Song, and T. Lan, "Live gradient compensation for evading stragglers in distributed learning," in *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, 2021, pp. 1–10.

[63] A. Reisizadeh, S. Prakash, R. Pedarsani, and A. S. Avestimehr, "Codedreduce: A fast and robust framework for gradient aggregation in distributed learning," *IEEE/ACM Transactions on Networking*, vol. 30, no. 1, pp. 148–161, 2022.

[64] B. Buyukates, E. Ozfatura, S. Ulukus, and D. Gündüz, "Gradient coding with dynamic clustering for straggler-tolerant distributed learning," *IEEE Transactions on Communications*, vol. 71, no. 6, pp. 3317–3332, 2023.

[65] Z. Chai, Y. Chen, A. Anwar, L. Zhao, Y. Cheng, and H. Rangwala, "Fedat: A high-performance and communication-efficient federated learning system with asynchronous tiers," in *SC21: International Conference for High Performance Computing, Networking, Storage and Analysis*, 2021, pp. 1–17.

[66] Y. Guan, Z. Yuan, G. Sun, and J. Cong, "Fpga-based accelerator for long short-term memory recurrent neural networks," in *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2017, pp. 629–634.

[67] D. Danopoulos, I. Stamoulias, G. Lentaris, D. Masouros, I. Kanaropoulos, A. K. Kakolyris, and D. Soudris, "Lstm acceleration with fpga and gpu devices for edge computing applications in b5g mec," in *International Conference on Embedded Computer Systems*. Springer, 2022, pp. 406–419.

[68] Huawei, "Microwave industry," *White paper*, 2023, Accessed: September 21, 2023. [Online]. Available: https://www-file.huawei.com/-/media/corp2020/pdf/tech-insights/1/2023_microwave_industry_white_paper.pdf?la=fr-fr

[69] N.-E.-H. Yellas, S. Boumerdassi, A. Ceselli, and S. Secci, "Complexity-performance trade-offs in robust access point clustering for edge computing," in *2021 17th International Conference on the Design of Reliable Communication Networks (DRCN)*, 2021, pp. 1–8.

[70] I. I. CPLEX, "Ibm ilog ampl version 12.2: User's guide," *New York*, 2010.

[71] I. I. Cplex, "V12. 1: User's manual for cplex," *International Business Machines Corporation*, vol. 46, no. 53, p. 157, 2009.

[72] F. Giust, G. Verin, K. Antevski, J. Chou, Y. Fang, W. Featherstone, F. Fontes, D. Frydman, A. Li, A. Manzalini *et al.*, "Mec deployments in 4g and evolution towards 5g," *ETSI White paper*, vol. 24, no. 2018, pp. 1–24, 2018.

[73] N.-E.-H. Yellas, S. Boumerdassi, A. Ceselli, B. Maaz, and S. Secci, "Robust access point clustering in edge computing resource optimization," *IEEE Transactions on Network and Service Management*, vol. 19, no. 3, pp. 2738–2750, 2022.

[74] S. Schwarzmann, C. C. Marquezan, R. Trivisonno, S. Nakajima, V. Barriac, and T. Zinner, "Ml-based qoe estimation in 5g networks using different regression techniques," *IEEE Transactions on Network and Service Management*, vol. 19, no. 3, pp. 3516–3532, 2022.

[75] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[76] A. H. Project, "D2.1: Use cases, requirements, and preliminary system architecture."

[77] T. Yang, G. Andrew, H. Eichner, H. Sun, W. Li, N. Kong, D. Ramage, and F. Beaufays, "Applied federated learning: Improving google keyboard query suggestions," *arXiv preprint arXiv:1812.02903*, 2018.

[78] N.-E.-H. Yellas, B. Addis, R. Riggio, and S. Secci, "Function placement and acceleration for in-network federated learning services," in *2022 18th International Conference on Network and Service Management (CNSM)*, 2022, pp. 212–218.

[79] "Kubernetes," Accessed: June 10, 2022. [Online]. Available: http://kubernetes.io.

[80] C.-D. Phung, N.-E.-H. Yellas, S. B. Ruba, and S. Secci, "An open dataset for beyond-5g data-driven network automation experiments," in *2022 1st International Conference on 6G Networking (6GNet)*, 2022, pp. 1–4.

[81] M. R. Sprague, A. Jalalirad, M. Scavuzzo, C. Capota, M. Neun, L. Do, and M. Kopp, "Asynchronous federated learning for geospatial applications," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2018, pp. 21–28.

[82] W. da Silva Coelho, A. Benhamiche, N. Perrot, and S. Secci, "Function splitting, isolation, and placement trade-offs in network slicing," *IEEE Transactions on Network and Service Management*, vol. 19, no. 2, pp. 1920–1936, 2022.

[83] S. Hosseinalipour, C. G. Brinton, V. Aggarwal, H. Dai, and M. Chiang, "From federated to fog learning: Distributed machine learning over heterogeneous wireless networks," *IEEE Communications Magazine*, vol. 58, no. 12, pp. 41–47, 2020.

[84] D. Justus, J. Brennan, S. Bonner, and A. S. McGough, "Predicting the computational cost of deep learning models," in *2018 IEEE International Conference on Big Data (Big Data)*, 2018, pp. 3873–3882.

[85] *https://github.com/nehyellas/IFLC*. Accessed: June 25, 2023. [Online]. Available: https://github.com/nehyellas/IFLC

[86] "Pwlf: Piecewise linear fit," Accessed: October 6, 2023. [Online]. Available: https://jekel.me/piecewise_linear_fit_py/

[87] ETSI, "5g; system architecture for the 5g system (5gs) (3gpp ts 23.501 version 16.7.0 release 16)," *ETSI, Sophia-Antipolis, France, Tech. Rep. GR MEC*, 2021.

Appendix A

Multi-objective Access Points Clustering

MIN-MAX:

$$\min \sum_{i\in\bar{A},j\in\bar{A}} c_{ij} * z_{ij}$$

$$c_{ij} = \max_{t\in T}|d_i^t - d_j^t| \qquad \forall i,j \in \bar{A}$$

$$\sum_{j\in\bar{A}} z_{ij} + z_{ji} = 1 \qquad \forall i \in \bar{A}$$

$$z_{ij} = 0 \qquad \forall i,j \in \bar{A} : \delta_{ij} > \tilde{\delta}$$

$$z_{ij} \in \{0,1\} \qquad \forall i,j \in \bar{A}$$

MIN-SUM:

$$\min \sum_{i\in\bar{A},j\in\bar{A}} c_{ij} * z_{ij}$$

$$c_{ij} = \frac{\sum_{t\in T}|d_i^t - d_j^t|}{\mathbf{card}(T)} \qquad \forall i,j \in \bar{A}$$

$$\sum_{j\in\bar{A}} z_{ij} + z_{ji} = 1 \qquad \forall i \in \bar{A}$$

$$z_{ij} = 0 \qquad \forall i,j \in \bar{A} : \delta_{ij} > \tilde{\delta}$$

$$z_{ij} \in \{0,1\} \qquad \forall i,j \in \bar{A}$$

MAX-MAX:

$$\max \sum_{i\in\bar{A},j\in\bar{A}} c_{ij} * z_{ij}$$

$$c_{ij} = \max_{t\in T}|d_i^t - d_j^t| \qquad \forall i,j \in \bar{A}$$

$$\sum_{j\in\bar{A}} z_{ij} + z_{ji} = 1 \qquad \forall i \in \bar{A}$$

$$z_{ij} = 0 \qquad \forall i,j \in \bar{A} : \delta_{ij} > \tilde{\delta}$$

$$z_{ij} \in \{0,1\} \qquad \forall i,j \in \bar{A}$$

MAX-SUM:

$$\max \sum_{i\in\bar{A},j\in\bar{A}} c_{ij} * z_{ij}$$

$$c_{ij} = \frac{\sum_{t\in T}|d_i^t - d_j^t|}{\mathbf{card}(T)} \qquad \forall i,j \in \bar{A}$$

$$\sum_{j\in\bar{A}} z_{ij} + z_{ji} = 1 \qquad \forall i \in \bar{A}$$

$$z_{ij} = 0 \qquad \forall i,j \in \bar{A} : \delta_{ij} > \tilde{\delta}$$

$$z_{ij} \in \{0,1\} \qquad \forall i,j \in \bar{A}$$

MIN-CORR:

$$\min \sum_{i\in\bar{A},j\in\bar{A}} c_{ij} * z_{ij}$$

$$c_{ij} = \max_{t\in T} \text{ratio}_{i\,j} \qquad \forall i,j \in \bar{A}$$

$$\text{ratio}_{i\,j} = \frac{R_{ij} * R_{ji}}{\sqrt{\sigma_i} * \sqrt{\sigma_j}} \qquad \forall i,j \in \bar{A}$$

$$R_{ij} = \sum_{t\in T} \frac{d_i^t - d_j^t}{2}, R_{ji} = \sum_{t\in T} \frac{d_j^t - d_i^t}{2} \qquad \forall i,j \in \bar{A}$$

$$\sigma_i = \sqrt{\sum_{t\in T}(d_i^t - \bar{d_i})^2}, \sigma_j = \sqrt{\sum_{t\in T}(d_j^t - \bar{d_j})^2} \qquad \forall i,j \in \bar{A}$$

$$\sum_{j\in\bar{A}} z_{ij} + z_{ji} = 1 \qquad \forall i \in \bar{A}$$

$$z_{ij} = 0 \qquad \forall i,j \in \bar{A} : \delta_{ij} > \tilde{\delta}$$

$$z_{ij} \in \{0,1\} \qquad \forall i,j \in \bar{A}$$

MAX-CORR:

$$\max \sum_{i\in\bar{A},j\in\bar{A}} c_{ij} * z_{ij}$$

$$c_{ij} = \max_{t\in T} \text{ratio}_{i\,j} \qquad \forall i,j \in \bar{A}$$

$$\text{ratio}_{i\,j} = \frac{R_{ij} * R_{ji}}{\sqrt{\sigma_i} * \sqrt{\sigma_j}} \qquad \forall i,j \in \bar{A}$$

$$R_{ij} = \sum_{t\in T} \frac{d_i^t - d_j^t}{2}, R_{ji} = \sum_{t\in T} \frac{d_j^t d_i^t}{2} \qquad \forall i,j \in \bar{A}$$

$$\sigma_i = \sqrt{\sum_{t\in T} (d_i^t - \bar{d}_i)^2}, \sigma_j = \sqrt{\sum_{t\in T} (d_j^t - \bar{d}_j)^2} \qquad \forall i,j \in \bar{A}$$

$$\sum_{j\in\bar{A}} z_{ij} + z_{ji} = 1 \qquad \forall i \in \bar{A}$$

$$z_{ij} = 0 \qquad \forall i,j \in \bar{A} : \delta_{ij} > \tilde{\delta}$$

$$z_{ij} \in \{0,1\} \qquad \forall i,j \in \bar{A}$$

MIN-CORR-VAR:

same as MIN-CORR, where we use $d_i'^t$ instead of d_i^t.

MAX-CORR-VAR:

same as MAX-CORR, where we use $d_i'^t$ instead of d_i^t.

Acronyms

3GPP 3rd Generation Partnership Project.

AI Artificial Intelligence.

AIF Artificial Intelligence Function.

AMPL A Mathematical Programming Language.

AP Access Point.

BBU Base Band Unit.

BS Base Station.

CDF Cumulative Distribution Function.

CO Central Offices.

CPU Central Processing Unit.

C-RAN Centralized Radio Access Network.

CU Centralized Unit.

DU Distributed Unit.

FL Federated Learning.

FPGA Field-Programmable Gate Array.

GPU Graphics Processing Unit.

HWA HardWare Acceleration.

if interface.

IFLC In-network Federated Learning Control.

ILP Integer Linear Programming.

KPI Key Performance Indicator.

LSTM Long-Short Term Memory.

MANO Management and Orchestration.

MEC Multi-access Edge Computing.

MILP Mixed-Integer Linear Programming.

ML Machine Learning.

MNO Mobile Network Operator.

NFV Network Function Virtualization.

NPU Neural Processing Unit.

NWDAF NetWork Data Analytics Function.

O-RAN Open RAN.

PoP Points of Presence.

QoE Quality of Experience.

QoS Quality of Service.

RAN Radio Access Network.

RRH Remote Radio Head.

RU Radio Unit.

SDN Software Defined Networking.

SLA Service Level Agreement.

UE User Equipment.

VM Virtual Machine.

VNF Virtual Network Function.

vRAN virtualized Radio Access Network.