A novel smart grid architecture that facilitates high RES penetration through innovative markets towards efficient interaction between advanced electricity grid management and intelligent stakeholders

H2020-GA-863876

Final Version of Distributed Flexibility Asset Markets, P2P Trading Models and Advanced Retail Market Mechanisms

Deliverable D3.3

Document Information

Scheduled delivery	30.11.2021
Actual delivery	30.11.2021
Version	Final
Responsible Partner	UCY

Dissemination Level

PU Public

Contributors

Prodromos Makris (ICCS), Nikolaos Efthymiopoulos (ICCS), Konstantinos Steriotis (ICCS), Maria-Iro Baka (UCY), Christina Papadimitriou (UCY), George Georghiou (UCY), Domagoj Badanjak (UNIZG-FER)

Internal Reviewers

Hrvoje Pandzic (UNIZG-FER), Emmanouel Varvarigos (ICCS)

Acknowledgements

The research leading to these results has received funding from the EC Framework Programme HORIZON2020/2014-2020 under grant agreement n° 863876.

Glossary of Acronyms

Project management terminology

Acronym	Definition
D	Deliverable
HLUC	High Level Use Case
MS	Milestone
WP	Work Package
UCS	Use Case Scenario

Technical terminology

Acronym	Definition
AA	Adjustable Assets
AC-OPF	Alternate Current Optimal Power Flow
AFAT	Automated Flexibility Aggregation Toolkit
AI	Artificial Intelligence
ARMM	Advanced Retail Market Mechanism
ATP	Automated Trading Platform
B2B/B2C	Business to Business / Business to Consumer
BRP	Balance Responsible Party
B-RTP	Behavioural Real Time Pricing
COMNET	Computing and Networking
DA	Day-Ahead
DC	Data Center
DER	Distributed Energy Resource
DFA	Distributed Flexibility Asset
DLFM	Distribution Level Flexibility Market
DR	Demand Response
DROaaS	Demand Response Operation as a Service
DSO/TSO	Distribution/Transmission System Operator
EC	Energy Community
EMS	Energy Management System
EV	Electric Vehicle
FPGA	Field-programmable gate array
GBP	Generic Business Process
GPU	Graphics processing unit
GUI	Graphical User Interface
HVAC	Heating, Ventilation and Air Conditioning
ICT	Information and Communication Technology
ILP	Integer Linear Program
KPI	Key Performance Indicator
LSTM	Long short-term memory
MAE	Mean Absolute Error

MDP	Markov Decision Process
ML	Machine Learning
MLP	Multi-Layer Perceptron
MTU	Market Time Unit
NN	Neural Network
OTC	Over-the-Counter
PDF	Probability Density Function
RES	Renewable Energy Sources
S/W	Software
SA	Shiftable Assets
SDN	Software Defined Networking
V2G	Vehicle to Grid

Table of Contents

List of Figures and Tables

List of Figures

List of Tables

Document History

This deliverable includes the final version of the mathematical models, research problem formulations, algorithms and performance evaluation results for the operation of the FLEXGRID's flexibility aggregation markets.

Table 1: Document History Summary

Revision Date	File version	Summary of Changes
07/07/2021	v0.1	Draft ToC circulated among all consortium partners
30/07/2021	v0.2	All partners commented on the draft ToC structure.
05/09/2021	v0.3	Final ToC version has been agreed and writing task delegations have been provided to ICCS and UCY.
30/09/2021	v0.4	ICCS provided input for the first version of chapters 3 and 4.
26/10/2021	v0.6	UCY provided input for the first version of chapter 2.
15/11/2021	v0.8	UCY integrated all contributions and pre-final D3.3 version has been sent to UNIZG-FER for internal review.
22/11/2021	v0.9	UNIZG-FER made a thorough review and requested for changes to enhance the quality of the deliverable.
29/11/2021	v0.95	ICCS and UCY addressed all comments from the internal review process and forwarded the final version to the coordinator.
30/11/2021	Final	Coordinator (ICCS) made final enhancements/changes and submitted to ECAS portal

Executive Summary

This report is an official deliverable of H2020-GA-863876 FLEXGRID project dealing with the detailed architecture design of all WP3 subsystems and their interactions as well as the respective technical specifications emphasizing on the detailed description of WP3 research problems. **The focus of this document is FLEXGRID High Level Use Case #4 (HLUC_04), which deals with the operation of automated flexibility aggregation as a service to independent aggregators.** Three Use Case Scenarios (UCSs) are presented for the optimization of the business portfolio of the aggregator, which consists of end energy users/prosumers and their flexibility assets. The respective algorithms will be integrated in a S/W toolkit called Automated Flexibility Aggregation Toolkit (AFAT), which will dynamically interact with the core FLEXGRID Automated Trading Platform (ATP) in the context of WP6.

Chapter 1 is an introduction to this report and summarizes the scope and purpose of this document. In detail, it provides a description of High-Level Use Case #4 and the interaction with the FLEXGRID system, summarizes the research innovations, the research impact on current and future aggregator's business models as well as policy recommendations and lessons learnt.

Chapters 2-4 are organized in a similar structure and present the WP3 research results that were not concluded in D3.1 and D3.2 in a coherent manner. Specifically, each chapter presents:

- A summary of FLEXGRID WP3 research results so far (from previous D3.1 and D3.2)
- System model
- Problem formulation
- Algorithmic solution
- Simulation setup and performance evaluation results
- Concluding remarks and lessons learned

Chapter 2 focuses on the research problem of the FLEXGRID UCS 4.1 and more specifically on the aggregator's optimization tool regarding the management of a FlexRequest. The aggregator's objective is to maximize its profits from participation in the flexibility market. This translates to maximization of its revenues and minimization of associated costs. The revenues of the aggregator stem from positive responses to FlexRequests. Costs are associated with end-user compensations for provision of flexibility, defined in FlexContracts and potential imbalance costs, meaning the costs of any imbalance/unwanted deviation from the scheduled operating pattern and requested flexibility activation.

Chapter 3 introduces a variation of UCS 4.1, where the aggregator needs to manage the flexibility of its portfolio when many sources of uncertainty are present. In order to deal with stochasticity, advanced scheduling techniques are proposed, which combine deep learning theory with duality theory. In this work, we assume a slightly different system model compared to chapter 2, in which the aggregator is responsible for her own imbalances and thus a scheduled operating pattern of flexibility assets is not known beforehand. In other

words, in chapter 2, the aggregator represents only the flexibility of the assets of the portfolio (participation only in flexibility markets - DLFM) and a supplier is responsible for the energy of the day-ahead scheduled operating pattern. On the other hand, in chapter 3, the aggregator is responsible for trading both energy and flexibility in existing TN-level markets as well as in the proposed DLFM including thus market-related stochasticity in the mathematical model.

Chapter 4 presents the research problem of the FLEXGRID UCS 4.2 entitled “Aggregator operates an ad-hoc B2C flexibility market with its end energy prosumers”. In this novel B2C flexibility market, we assume that the end users compete with each other to provide flexibility services to the aggregator. The details of each end user’s utility function are stated via the FlexContract that is agreed with the aggregator. In particular, we draw on concepts of mechanism design theory in order to define an iterative, auction-based mechanism, consisting of an allocation rule and a payment rule. Through the auction procedure, the aggregator exchanges messages with the end users in the form of queries. A query in our case is a price signal communicated from the aggregator to the end user, to which the end user responds with his/her preferred action (e.g. consumption reduction) according to this signal. This problem becomes very difficult to solve when we consider a large number of FlexRequests published by the FLEXGRID ATP, a large portfolio of end users (i.e. at a scale of several hundreds of thousands of end users or even millions), more complex (and thus more realistic) FlexAsset models and more stringent real-time constraints imposed by the emerging B2C/B2B flexibility markets. As a result, we propose an optimal cloud resource allocation algorithm, which is able to service multiple FlexRequests (e.g. in multiple distribution networks), and minimize the cost of computational resources, while respecting the execution time constraints of each FlexRequest. This will help towards a cost-efficient and competitive B2C flexibility market as a service offering by the aggregator.

Conclusively, in Chapter 5, we describe the integration of the research work of WP3 within the AFAT and FLEXGRID ATP (WP6), the validation and implementation in pilot sites (WP7) and the relation/interaction of the aggregator within the WP8 business models, values propositions, and the exploitation plan.

1 Introduction

1.1 Description of High-Level Use Case #4 and interaction with the FLEXGRID system as a whole

High Level Use Case (HLUC) #4 of FLEXGRID revolves around the role of the aggregator and especially on the role of the flexibility aggregator. Flexibility aggregators are considered as market actors, which aggregate flexibility/demand from energy prosumers and/or consumers and represent the aggregated portfolio in markets as flexibility providers. The demand side of the flexibility market, flexibility buyers, are different stakeholders such as TSOs, DSOs and BRPs, which opt to purchase flexibility from different market mechanisms.

The flexibility aggregator market actor can participate in various electricity markets and interacts with both market participants and end energy prosumers. The focus of HLUC #4 focuses on variations of the aggregator's business model under different market designs, the interaction with end-user for the construction of the portfolio and the representation and orchestration of the aggregated flexibility portfolio in several markets (both existing TN-level ones and the DLFM proposed by FLEXGRID). The main purpose of this HLUC is the operation of automated flexibility aggregation for optimal use of available flexibility in the distribution level and the development of sustainable and profitable business models for the aggregator entity itself and participants of the portfolio. Within this HLUC, different approaches were examined, which have led to the development of different mathematical models and algorithms to be applied for the optimal use of distributed flexibility assets (DFAs). These Use Case Scenarios (UCSs) are documented and described in detail in previous deliverables, D2.1 and D2.2 (in Month 4 and 6 respectively) and elaborated work was done within WP3 of FLEXGRID.

In the previous WP3 deliverables (D3.1 and D3.2), three research problems (one per UCS of HLUC #4) have been clearly defined. In D3.1, a high-level description of the three use-cases/problems has been performed with related works from international literature. FLEXGRID's research contributions have been clearly defined and hints about the problem formulation, algorithmic solution, datasets to be used for the system-level simulations and most important key performance indicators (KPIs) have been presented. The work of D3.2 elaborates on the work of D3.1 by presenting close to final versions of mathematical modeling and proposed algorithms for all three of the research problems (final version of UCS4.3), along with available initial performance evaluation results.

This deliverable extends the work of the previous deliverables (D3.1 and D3.2) with the final version mathematical modeling of two of the research problems (UCS 4.1 and UCS 4.2) and the presentation of performance evaluation results considering realistic case-studies and the use of realistic datasets following the FLEXGRID data management plan (DMP).

1.2 Summary of FLEXGRID's research innovations

Following the survey work in the previous deliverables, three research problems involving the role of the flexibility aggregator were identified. For each of these research problems, mathematical models were formulated, and appropriate tools were developed to address the aggregator's needs and coordinate the required actions within the portfolio. The three research problems are the following:

1) The aggregator efficiently responds to FlexRequests made by TSO/DSO/BRPs by optimally orchestrating its aggregated flexibility portfolio of end energy prosumers (cf. UCS 4.1)
2) The aggregator operates an ad-hoc B2C flexibility market with its end energy prosumers by employing advanced pricing models and auction-based mechanisms (cf. UCS 4.2)
3) The aggregator wants to maximize its revenues by dynamically orchestrating its distributed FlexAssets from its end users to optimally participate in near-real-time energy/flexibility markets (cf. UCS 4.3)

Two variations of the first research problem are described in detail in chapters 2 and 3 respectively. The first variation presented in chapter 2 focuses on the orchestration of the aggregator's portfolio when the scheduled operating pattern of the distributed flexibility assets is known (i.e. deterministic approach), while the one presented in chapter 3 introduces sources of uncertainty regarding the operation of assets of the portfolio (i.e. stochastic approach). The second research problem described in chapter 4 focuses on the development of a novel B2C flexibility market architecture. In this B2C flexibility market, which is operated by an aggregator company, various types of small-scale distributed Flexibility Assets (DFAs) compete for the required aggregated flexibility to be gathered by the aggregator (i.e., FlexSupplier) at the least possible cost. The final mathematical model version and results of the third research problem, related to the dynamic orchestration of FlexAssets to participate in near-real-time energy/flexibility markets was described in detail in D3.2. For the remaining research problems, each chapter presents:

- A summary of FLEXGRID research results so far
- System model
- Problem formulation
- Simulation setup and performance evaluation results
- Concluding remarks and lessons learned

1.3 Summary of FLEXGRID's research impact on today and future aggregator's business

The work of WP3 focuses on the scientific excellence of the proposed FLEXGRID services (identified open research items) at TRL 3. The most important WP3 scientific results are adapted in order be able to serve the business needs of an aggregator. Thus, in WP6, our focus is on FLEXGRID's research impact on today and future aggregator's business by

demonstrating WP3 intelligence in the FLEXGRID ATP (i.e. TRL 5), which will be demonstrated and validated in WP7.

More specifically, AFAT's frontend (GUI)[1] will be comprised of three basic tabs, namely:

- Manage a FlexRequest (UCS 4.1)
- Create a FlexOffer (UCS 4.3)
- Manage a B2C flexibility market (UCS 4.2)

Following up the AFAT's frontend services, three main WP3 algorithms will be integrated in AFAT's backend, namely:

- A flexibility aggregation algorithm that explores control actions to identify the optimal flexibility dispatch order within a portfolio of available FlexAssets in order to respond to a given FlexRequest. The proposed solution is based on a centralized optimization model and it is described in detail in chapter 2 below (cf. UCS 4.1).
- A flexibility aggregation algorithm that automatically aggregates availability flexibility in price/quantity tuples for a given future timeframe. Then, the aggregated FlexOffer curve is automatically submitted in FLEXGRID ATP, so that it can be subsequently matched with a FlexRequest. The proposed solution is extensively described in chapter 3 of D3.2 (cf. UCS 4.3).
- A retail pricing algorithm via which the aggregator can run a novel B2C flexibility market. The proposed solution is based on a decentralized optimization model and it is described in chapter 4 below (cf. UCS 4.2).

Table 2 below summarizes how the WP3 research results (TRL 3) will be further exploited in WPs 6 and 8.

Table 2: Summary of interactions between WP3 research work (scientific excellence at TRL 3) and WP6/WP8 work about potential business impact

AFAT GUI (WP6)	Mode of operation	Business goal (WP8)
Manage a FlexRequest	Online	A new FlexRequest is published in real-time by a FlexBuyer in the ATP. The aggregator is instantly informed and then runs the UCS 4.1 algorithm to decide the updated dispatch per FlexAsset/end user that belongs to its portfolio.
	Offline	The aggregator performs "what-if" simulation scenarios (i.e. different configurations of FlexContracts, expansion/modification of portfolio, different sequence of FlexRequests, etc.) to determine strategies for optimal response to future FlexRequests. For a sequence of multiple FlexRequests assumed in a given "what-if" simulation scenario configured by the aggregator user, the UCS 4.1 algorithm will run iteratively.

[1] AFAT's frontend services (GUI) are being developed by ETRA within WP6 context.

Create a FlexOffer	Online	The aggregator creates a FlexOffer in real-time (in order to submit it in the ATP) based on the current availability of FlexAssets (cf. FlexContract per FlexAsset that denotes the available reserve capacity).
	Offline	The aggregator runs "what-if" scenarios to see whether it is more beneficial to participate in the existing TN-level balancing market or DN-level flexibility market (i.e. DLFM).
Manage a B2C flexibility market	Offline	The aggregator runs various "what-if" simulation scenarios via running an advanced retail pricing algorithm (Behavioral Real Time Pricing – B-RTP) to identify how it can recommend a new (more beneficial) FlexContract to a set of end energy prosumers.

The NODES market paradigm and platform setup are followed for the integration of WP3 research results. Technical and software development issues are supported by NODES real-life business experience, while NPC supports with its consultancy services regarding the integration of the proposed flexibility marketplace in the existing EU markets and regulations.

1.4 Summary of FLEXGRID policy recommendations and lessons learned

The research work of WP3 proposes novel mechanisms and optimization tools to facilitate the participation of the flexibility aggregator in current and future electricity market designs. An online flexibility marketplace (i.e. FLEXGRID ATP) is assumed, in which the aggregator acts as a flexibility provider (i.e. from FlexSupply side of the proposed DLFM). The aggregator can use the proposed set of intelligent mathematical models and algorithms to automate and dynamically adapt the flexibility aggregation process and qualify for market participation with its aggregated flexibility portfolio. The appropriate construction of FlexContracts with end-users and the proposed interactions with market participants can effectively lead to an active involvement of distributed flexibility assets in the market and facilitate the use of otherwise unexploited sources of flexibility to alleviate the flexibility-related problems in the current market designs. Based on the results and the multiple extensions of the research in WP3, policy and regulation makers can extend and formulate new options and market rules to enable the integration of the emerging market actor, the flexibility aggregator.

2 An aggregator efficiently responds to FlexRequests made by TSO/DSO/BRPs by optimally orchestrating its aggregated flexibility portfolio of end energy prosumers

The focus of this chapter is the research problem of the FLEXGRID's HLUC_04_UCS_01. In this specific Use Case Scenario (UCS), the aggregator needs to represent the flexibility of its portfolio of DERs (i.e. FlexAssets) in the market and manage the aggregated flexibility in a centralized manner.

2.1 Summary of FLEXGRID research results so far

In FLEXGRID UCS 4.1, the focus is on the role of the independent aggregator, as defined in the Clean Energy for all Europeans package (Directive (EU) 2019/944), which represents the flexible energy of the assets of its portfolio. The interaction of the aggregator with the existing and developing stages of the electricity market (flexibility buyers) is via FlexRequests, which as discussed in D3.2 are distinguished into two categories: Reserve and Dispatch. Rewards for prosumers (flexibility providers) which enlist their flexible assets in the aggregator's portfolio are determined based on FlexContracts (Figure 1). As discussed in previous deliverables (D3.1 and D3.2), **the main objective of an aggregator is to maximize its profit while ensuring a profitable business model for all participating end-users, respecting reservations of flexibility and avoiding any deviations from its targeted flexibility schedule.**

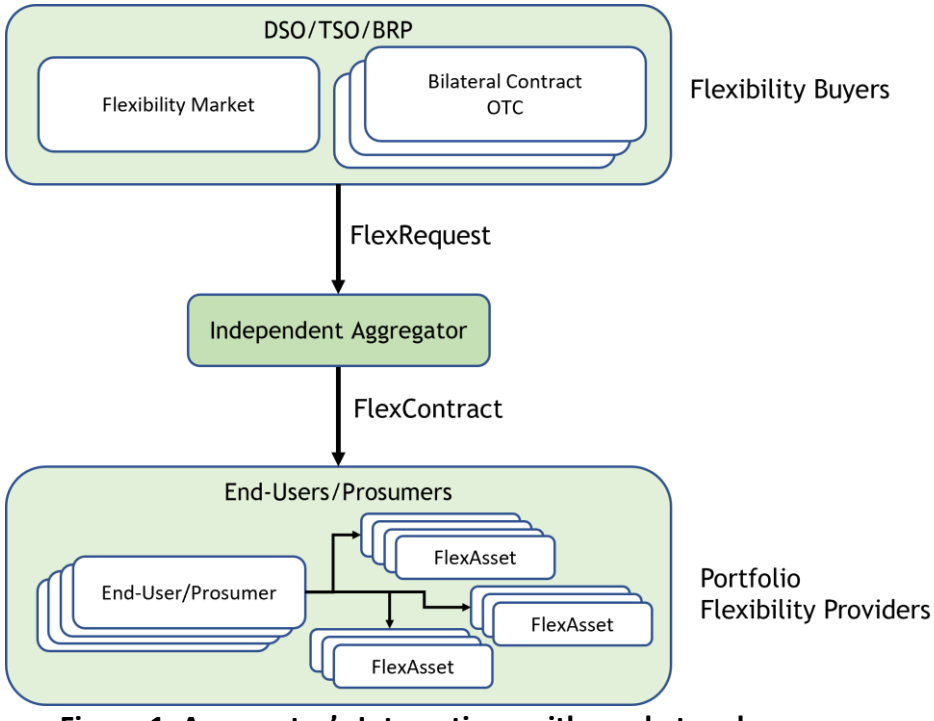

Figure 1: Aggregator's Interactions with market and prosumers

The work of the period M19-M26 documented in this deliverable is focused on providing a tool for the aggregator to "manage a FlexRequest". This requires the orchestration of its portfolio in order to comply with the accepted flexibility activations requested by the market and deciding on the operation/dispatch of relevant and suitable assets of the portfolio for all timeslots/Market Time Units (MTUs) of the time horizon.

The method developed in FLEXGRID for the aggregator to "manage a FlexRequest", provides the aggregator with an appropriate tool to decide at run-time the dispatch orders for its portfolio. The aggregator is able to maximize its profit (revenues minus costs), while at the same time respecting constraints imposed by the market and portfolio participants. Furthermore, the approach followed within FLEXGRID allows the aggregator to:

- Manage its portfolio to resemble and act as a single dispatchable unit, qualifying for market participation.
- Ensure profit for all participating end-users of its portfolio where fairness among the assets of end-users is secured with respect to their preferences and comfort levels.
- Participate in a plethora of future market designs with a diverse and widely applicable tool.

2.2 System model

The aggregator is responsible for the management of the flexibility (flexible operation) of a portfolio of flexible assets. Flexibility is defined as the activated (controlled) deviation of the operation from the scheduled operating pattern. The scheduled operating pattern is determined by the supplier of the flexible assets and is submitted to the independent aggregator. In the current European market design, this refers to energy purchases and commitments in the Day-Ahead (DA) market which correspond to the supply of the planned operation of the flexibility assets. The time horizon coincides with the known scheduled operating pattern provided by the DA market and is a 24-hour period divided into discrete hourly timeslots, MTUs.

2.2.1 Flexibility Requests

As stated above, the aggregator interacts with the electricity market by responding to flexibility requests. Flexibility requests can involve either availability (reserve type) or energy (dispatch type). The aggregator needs to deviate the operation (activate flexibility) of FlexAssets and orchestrate its portfolio when positively responding to a FlexRequest of the dispatch type. The possible sequences of FlexRequests of reserve and dispatch type are shown in Figure 2. The optimization tool described in this work focuses on the management of FlexRequests of the dispatch type.

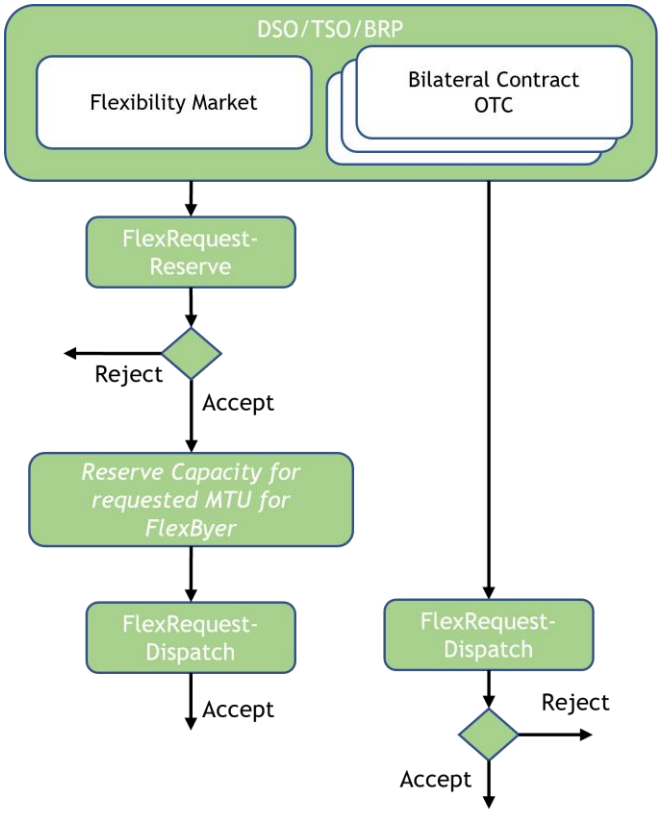

Figure 2: Diagram sequence of FlexRequests

A FlexRequest-Dispatch has the following attributes regarding the requested flexibility:

- Volume of requested energy (Up/Down)
- Direction of regulation (Upwards/Downwards)
- Reward
- Timetarget (single MTU)
- Location

Additionally, the timestamp of a FlexRequest is needed, which signifies both the time where the need of flexibility is discovered and when it becomes known to the aggregator.

2.2.2 Aggregator's Portfolio

The aggregator's portfolio consists of flexibility assets that are a subset of the assets represented by the supplier. An asset can be characterized and qualify as a flexibility asset when its operation can be controlled by the aggregator and the planned deviation of energy (flexibility activation/behavior) is linked and directly correlated to the aggregator control actions and signals. The aggregator registers end-users and their flexible assets to the portfolio via FlexContracts. End-user/asset compensation is defined within a FlexContract, along with users' preferences and constraints regarding the flexibility of each asset. Depending on the amount of flexibility that end-users offer to the portfolio (e.g., number of flexible assets, offered flexibility, flexibility behavior), each participant is rewarded with a reservation/participation reward which ensures profit when contributing to the aggregator's

portfolio. The reservation reward is determined upon the registration of the end-user and does not affect the decision process of the aggregator when managing a FlexRequest.

A second component of the end-user compensation, the dispatch/activation component, involves the actual activation of flexibility. The dispatch/activation component can be related either to the dispatched flexibility/energy (energy-dependent), depend on the number of flexibility activations within the time horizon or be based on the delay/advance of scheduled operation. This component of end-user compensation needs to be taken into account during the decision process of the aggregator regarding dispatch of assets and responses to FlexRequests.

2.2.2.1. Flexibility Assets

The aggregator represents a set of Shiftable Assets (SA) and a set of Adjustable Assets (AA), which are identified by a unique id and their owner, a corresponding end-user.

Shiftable Assets

Assets with shiftable operation patterns are categorized as Shiftable Assets. The control of assets in this category allows shifting the scheduled operation to a previous or future timeslot than one initially scheduled (scheduled operation is planned ahead and thus can be shifted to previous or next timeslots).

The scheduled operating pattern of each shiftable asset in SA is described by:

- Operation Pattern
- Scheduled start time

Information of shiftable assets regarding their flexibility behavior are:

- Minimum/earliest start time
- Maximum/latest start time or completion time
- Required activation notice
- Cost function

Additionally, the location (grid-based) of each shiftable asset is known, as it is required to determine the suitability of the asset for participating in the fulfilment of a FlexRequest.

Adjustable Assets

Adjustable assets are the ones with the ability to activate their flexibility based only on their scheduled operation pattern, without depending on the operation in previous timeslots and without affecting operation in future ones. The flexibility they offer to the aggregator's portfolio is defined by both the technical characteristics and the user constraints and preferences.

The scheduled operating pattern of each adjustable asset is given by the ratio of operation (percent of maximum operation) at each timeslot of the time horizon and the power of the asset.

Information of adjustable assets regarding their flexibility behavior are:

- Minimum operation
- Maximum operation
- Cost for upwards flexibility (increase generation/decrease consumption)
- Cost for downwards flexibility (decrease generation/increase consumption)
- Required activation notice
- Max total energy activation for the time window (User comfort/constraint)

Additionally, the location (grid-based) of each adjustable asset is known, as it is required to determine the suitability of the asset for participating in the fulfilment of a FlexRequest.

2.3 Problem Formulation

As stated above, the need for flexibility activation is not known in advance. FlexRequests of the dispatch type become known during the time horizon, which is indicated by the different timestamps. If all FlexRequests of the dispatch type were known in advance (prior to the time horizon), the aggregator would decide on the response of FlexRequests with all the information regarding the time horizon available and accordingly schedule its portfolio optimally based on the scheduled operation of its assets and the requested flexibility of accepted FlexRequests.

FlexRequests-dispatch are published at different timeslots and the aggregator needs to decide on the appropriate actions without any knowledge of future FlexRequests. As shown in Figure 2, when a FlexRequest-Dispatch succeeds a FlexRequest-Reserve, the aggregator is obliged to activate the requested flexibility. In a direct request, the aggregator can decide on its response depending on the reward of the FlexRequest and the status of its portfolio (availability and cost).

Activation of flexibility from shiftable assets (delay or advance of scheduled operation) does not cause variations on the total energy consumed/generated by a shiftable asset. However, as the operation pattern of shiftable assets involves more than a single MTU, any deviation of the scheduled operation affects multiple timeslots (Figure 3). Since a FlexRequest-dispatch involves flexibility activation for a single MTU, shiftable assets alone are not suitable for serving single FlexRequests. Negative values of deviating energy correspond to upwards regulation, decrease of consumption/increase of generation. Correspondingly, positive values of deviated energy correspond to downwards regulation, increase of consumption/decrease of generation.

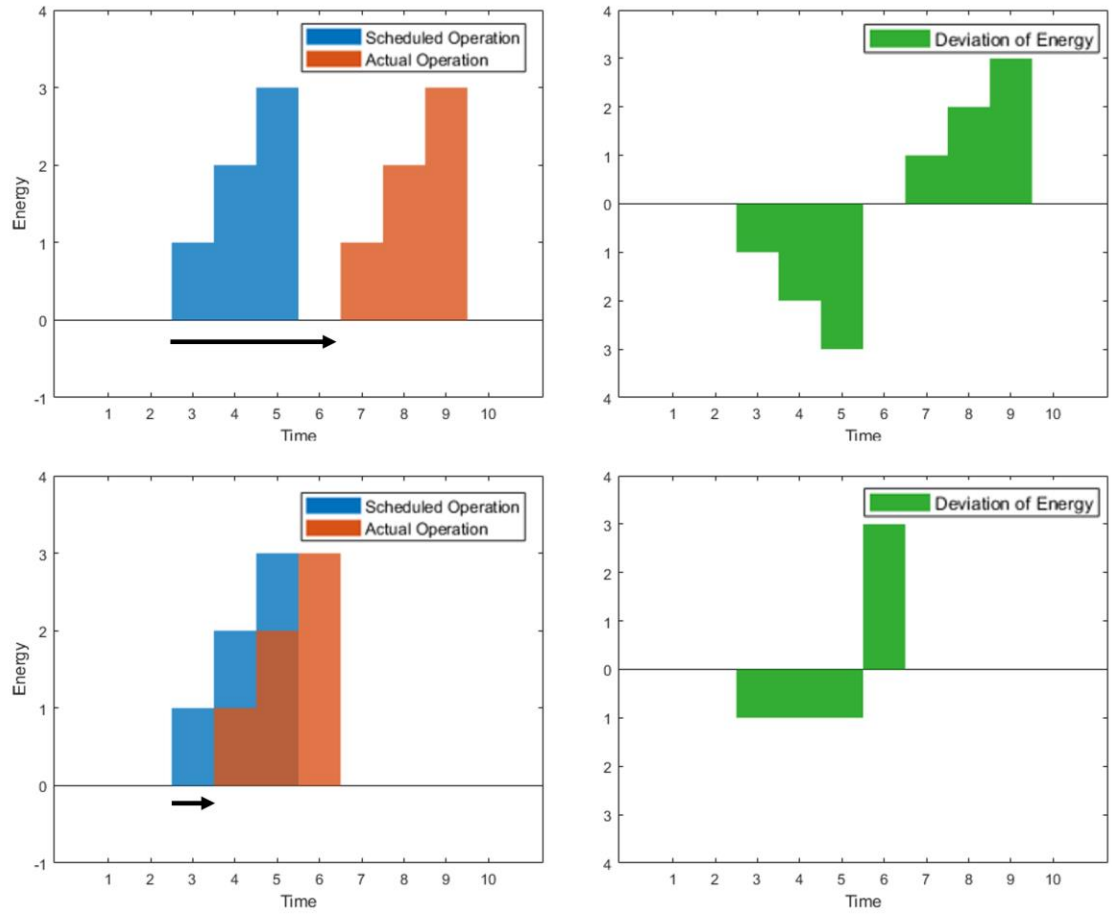

Figure 3: Deviation of operation of shiftable asset and deviation of energy

On the contrary, the aggregator is able to control the operation of adjustable assets for each individual MTU depending on the available flexibility and with respect to user preferences and constraints (Figure 4). The aggregator's control actions may alter the total energy consumed/generated by the adjustable asset and the potential consequences of these actions are included in the cost function of the adjustable asset.

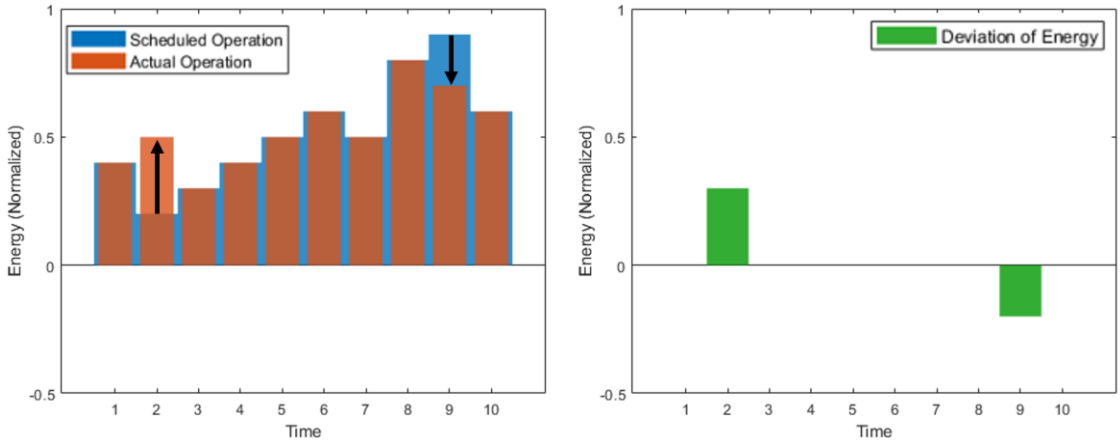

Figure 4: Deviation of operation of adjustable asset and deviation of energy

The cost of shifting the operation of a shiftable asset can be lower than the cost of adjusting the operation of an adjustable asset. In principle, flexibility from shiftable assets is

less costly, as it is assumed that the comfort level of end-users is affected less than during the adjustment of the operation of an adjustable asset. The optimal strategy for the aggregator is to satisfy requested flexible energy from multiple FlexRequests with a modification of the scheduled operation of a single shiftable asset. The objective is to use the effect of the shifted operation in multiple timeslots to the aggregator's advantage and cover any undesired deviations (deviations not requested by a FlexRequest) by orchestrating the operation of adjustable assets. This objective cannot always be reached, as not all FlexRequests are known beforehand and by the time all flexibility needs become known, the activation notices and/or scheduled start times may not allow control of suitable shiftable assets.

Additional options for the aggregator to manage any imbalances of the actions over his portfolio is to purchase energy from other market participants or to accept imbalance costs imposed by the market. As these options are expected to be less cost effective, the priority is to absorb undesired deviations within his own portfolio.

The feasibility of the solution, within the constraint of the network, can be included by providing specific grid locations and involving only the assets of the portfolio which are in accepted grid locations.

2.4 Algorithmic solution

The framework developed for this approach uses a mixed-integer linear programming (MILP) formulation, as it is a flexible and powerful method for solving large, complex problems. The objective function is the minimization of the cost of the aggregator as shown in equation 2.1:

$$\min_x C^T x = \min_x \left[C_{SA}, C_{AAup}, C_{AAdown,}, -R_{FRd}, C_{im}\right]^T \left[x_{SA}, x_{AAup}, x_{AAdown}, x_{FRd}, x_{im}\right] \quad (2.1)$$

subject to the following equality and inequality constraints:

$$\begin{aligned} Ax &\leq b \\ A_{eq}x &= b_{eq} \\ b_{low} &\leq x \leq b_{up} \end{aligned} \quad (2.2)$$

The decision and cost variables are distinguished into four categories:

- Shiftable assets
- Adjustable assets
- FlexRequests-Dispatch
- Imbalances

Shiftable Assets

Each shiftable asset is described with all possible operation patterns within the time horizon (all possible start times, considering the activation notice required). In turn, each operation pattern is assigned with a cost, which is derived by the cost function of the asset. The scheduled operating pattern is assigned a zero cost. Decision variables $x_{SA,j}$ correspond to the operation options of each shiftable asset j. These decision variables can take the value of 0 or 1 (integer variables) and as only one operation pattern can be selected for each shiftable asset j, only one of the corresponding decision variables can be 1.

Adjustable Assets

Adjustable assets can contribute to upwards or downwards regulation, depending on their scheduled operation pattern and the constraints imposed by the end-users. Two decision variables are assigned for the deviation of operation of each adjustable asset for all MTUs considering the activation notice of each asset. One for upwards regulation and one for downwards (x_{AAup}, x_{AAdown}). The deviation at each timeslot is limited by the technical and user constraints and the total deviation for the time horizon cannot surpass the limit indicated by the user.

FlexRequests and Imbalances

A decision variable is assigned to each FlexRequest and the corresponding cost equals to the negative reward, as the objective function is formulated as a minimization problem. Without loss of generality, a FlexRequest can either be fully accepted or rejected, thus the decision variable are integers which can take the value of 0 or 1.

A single decision variable corresponds to imbalances. The cost of imbalances represents the procurement of required energy out of the portfolio, from other market participants, or imbalance costs imposed by the market.

2.4.1. Continuous operation for the time horizon

The mixed-integer linear programming problem is performed for every timeslot (MTU) of the time horizon, where the number of cost and decision variables depend on the available information and control actions.

The options of decision variables for FlexRequests that have been accepted in a previous timeslot, but involve a future MTU, or that are an obligation for the aggregator, follow accepted FlexRequests-Reserve, are limited to acceptance and both lower and upper bounds are set to 1. Any other FlexRequests that involve future timeslots and are available at the current MTU can be either rejected (0) or fully accepted (1).

The actual values of decision variables regarding the portfolio ($x_{SA}, x_{AAup}, x_{AAdown}$) and the imbalances ($x_{im}$) become final when the aggregator can no longer perform any control

actions due to planned operation time and activation notice. Up to that point, the aggregator searches for alternative options to satisfy FlexRequests.

2.4.2. Fairness among participating end-users

The profit of end-users is ensured by the participation reward and the customized cost function for each of their participating assets. The aggregator opts to prioritize the activation of economic resources/assets, which is in line with the aggregator's business model and the fair treatment of end-users. Fairness, as an issue, arises among assets with the same cost for a given timeslot. In that case, the aggregator should not favor one asset over another. The activation of flexibility is distributed among the assets with the same or similar cost, with respect to user constraints, availability on future timeslots and future requirements.

2.5 Simulation setup and performance evaluation results

At the time of this deliverable, there are limited operational flexibility markets, with most of them being in a pilot/test phase, thus realistic datasets for flexibility requests are quite hard to come across. Regulating prices and volumes provide some information on the flexibility needs of TSOs, but not for other potential flexibility buyers (DSOs and BRPs) which are relevant for the independent aggregator's business model.

The research goal at this stage, is to prove that a portfolio of FlexAssets can be coordinated and orchestrated to act as a single dispatchable unit, to qualify for participation in the electricity markets, capable of offering balancing and other ancillary services. For this reason, synthetic data are suitable and used to represent the flexibility portfolio and the FlexRequests.

2.5.1. Simulation setup

The portfolio of the aggregator is assumed to consist of 10 end-users, 7 households and 3 small enterprises, where each end-user contributes to the flexibility portfolio with 2-3 shiftable assets and 1-2 adjustable assets. Without loss of generality, all assets are considered to have solely consumption patterns and all end-users are assumed to be a subset of the aggregator's portfolio suitably located within the grid for responding to and serving the of received FlexRequests. The time horizon is a single day divided into 24 hourly timeslots/MTUs.

Portfolio

For flexibility assets of the portfolio:

- Scheduled operation takes place within the entire time horizon to align with different types of residential and business consumption profiles
- Deviation options of shiftable assets are limited according to the users' preferences and always fall within the interval [-10,10] with the cost set to be higher for largest absolute values of deviations

- Maximum deviation of adjustable assets is within ±30% of the nominal energy consumption, depending on availability and can be limited by users' preferences. The cost of flexibility activation follows a tiered pricing scheme
- The maximum flexibility activation of adjustable assets for the time horizon is randomly selected within the interval of [3-5] of maximum activations, while on a different scenario, the maximum flexibility activation is relaxed.
- Required activation notice of assets is randomly selected within the interval [1-3]

Available flexibility of portfolio

The scheduled operation/consumption pattern of flexibility assets of the portfolio is shown in Figure 5. Both types of distributed assets of the portfolio, shiftable and adjustable assets provide flexibility to the portfolio. The available flexibility stemming from each type of asset however cannot be measured by the same metric. Flexibility due to the adjustable assets can be easily measured by the available reduction and increase of their consumption pattern at each MTU. The two dotted lines in Figure 5 indicate the maximum and minimum operation of adjustable assets.

Measuring and evaluating the available flexibility of shiftable assets however is not straightforward. Multiple options of rescheduling their operation leads to various possibilities of combinations of upwards and downward flexibility activations. As the shiftable assets of the portfolio are consumption assets, their planned operation limits the available options for upwards regulation. In Figure 5, the scheduled operation of shiftable assets is illustrated, which is indicative of the maximum upwards regulation that shiftable assets can offer.

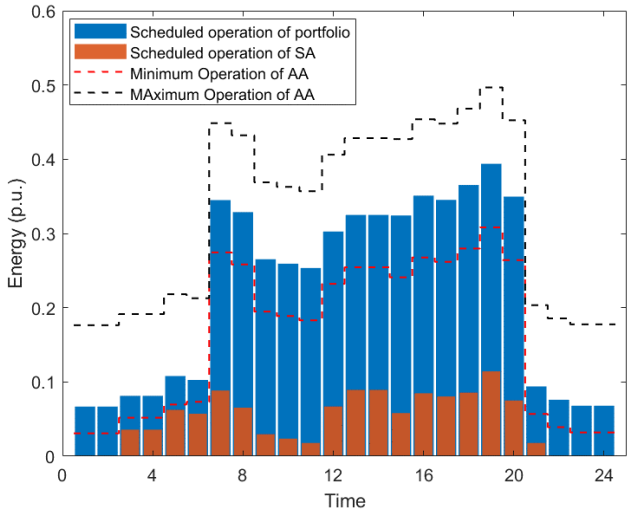

Figure 5: Scheduled operation pattern of portfolio

Flexibility Requests

FlexRequests received from the market are adjusted to the size of the portfolio. Different scenarios of energy requests from the market, within the range of [0-10%], [0-20%] and [0-

30%] of the scheduled operating pattern of the assets of the portfolio. In a pool-organized market, there is a single FlexRequest for each MTU in one direction, up or down. In a different market organization (continuous trading, OTC, bilateral contracts), there can be multiple FlexRequests towards the aggregator in different directions depending on the needs, obligations and requirements of each market participant. For the purpose of this research, we consider that the aggregator receives maximum two FlexRequests for each MTU and that the FlexRequests are always in the same direction to exclude profit from arbitrage. The monetary reward per energy unit is taken as constant for the entire time horizon. Although the price for flexible energy variates throughout the day, the goal is to showcase the trends of orchestrating a portfolio consisting of multiple distributed flexibility assets (DFAs). The distance of timestamps and the time targets of FlexRequests is set to 1 to illustrate the unforeseen and unpredictable need for flexibility.

As stated above, the portfolio consists of consumption assets, leading to the focus of the results on flexibility activation regarding upwards flexibility, meaning reduction of consumption. From the Nordpool market, and specifically from the regulating power, it is evident that there are needs for both upwards and downwards regulation. Due to the market operation and the nature of the aggregator's portfolio, accepting requests for downwards regulation is rarely profitable for the aggregator. This will be further analyzed in section 2.6.

2.5.2. Benchmark scenarios

The proposed approach is compared with two scenarios which are considered benchmarks. The first benchmark (Benchmark 1), which serves as the theoretical upper bound of the optimization/decision problem, is the one where all FlexRequests are known a priori (before the time horizon). This allows the aggregator to deterministically decide on the dispatch order and schedule of all flexibility assets.

The second scenario (Benchmark 2) is a more conservative approach, where the aggregator decides for each MTU, avoiding deviations in MTUs where flexibility is not requested, thus limiting its portfolio to adjustable assets which can be seen as more conventional sources of flexibility.

2.5.3. Performance evaluation results

In this section, the simulation results and their performance evaluation are presented. The acceptance ratio of FlexRequests when the aggregator receives requests for upwards regulation, both in terms of activated energy and number of positive responses, of the proposed approach and the two benchmark scenarios is shown in Table 3 for multiple ranges of requested energy from FlexRequests.

Table 3: Acceptance Ratio of upward FlexRequests

	Range of energy of FlexRequests: [0-10%]		
	Proposed Approach	**Benchmark 1**	**Benchmark 2**
Acceptance Ratio of FlexRequests in terms of energy	21.79%	22.5%	18.62%

Acceptance Ratio (Positive responses to Flex Requests)	25%	37.5%	31.25%
	Range of energy of FlexRequests: [0-20%]		
	Proposed Approach	**Benchmark 1**	**Benchmark 2**
Acceptance Ratio of FlexRequests in terms of energy	7.41%	11.55%	5.91%
Acceptance Ratio (Positive responses to Flex Requests)	14.58%	18.75%	12.5%
	Range of energy of FlexRequests: [0-30%]		
	Proposed Approach	**Benchmark 1**	**Benchmark 2**
Acceptance Ratio of FlexRequests in terms of energy	6.45%	7.37%	4.65%
Acceptance Ratio (Positive responses to Flex Requests)	22.92%	22.92%	22.92%

Benchmark 1 sets the upper limit for positive responses to FlexRequests. It should be noted, that when energy is rewarded with the same price throughout the time horizon, the acceptance ratio in terms of energy is more relevant than the acceptance ratio based on the absolute number of accepted FlexRequests. This explains the higher acceptance ratio in responses to FlexRequests in certain cases for Benchmark 2. An interesting observation are the similar values of acceptance ratio of the proposed approach and the two benchmarks for larger requests of energy. This is since the activated flexible energy reaches the limit of the total available flexible energy of the portfolio and more information and optimized approaches cannot significantly improve the acceptance ratio.

In terms of profit, the proposed approach achieves 78%-96% of the performance of benchmark 1 and outperforms benchmark 2 by over 6%-35% as shown in Figure 6. It should be noted that the difference in performance in terms of profit is larger than the difference in performance in terms of acceptance ratio. Although the aggregator cannot increase the number of positive responses, in the proposed approach and in benchmark 1, due to limited available flexibility, the dispatch order is optimized leading to larger profits.

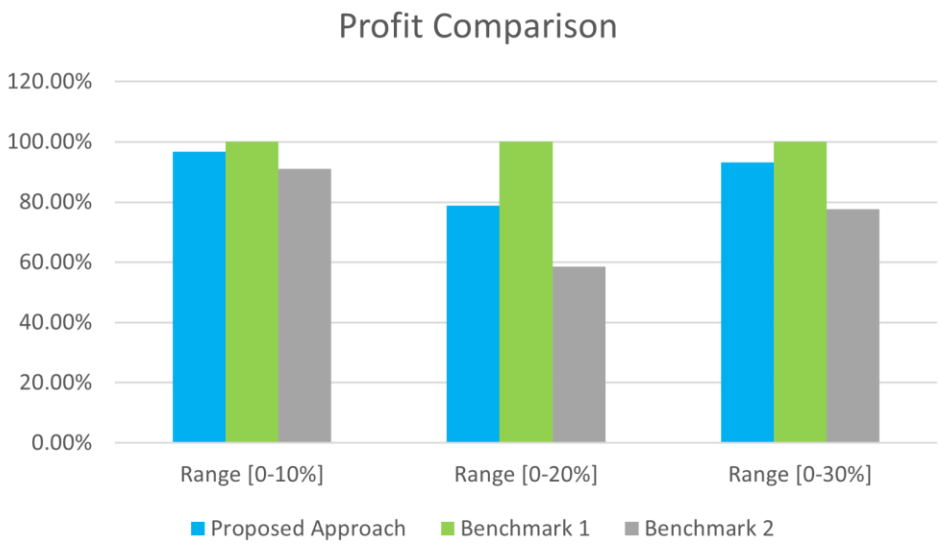

Figure 6: Comparison of profits of the proposed approach and the two benchmarks

Available flexibility of the portfolio is limited mainly due to two reasons. The first one involves the minimum and maximum operation of adjustable assets and the second one the maximum activation for each flexibility asset. To understand the effect of limited flexibility, the maximum activation of assets was relaxed. In Table 4 and Figure 7, the corresponding results for acceptance ratio and profits of the proposed approach and the two benchmarks, with a relaxation of the limit of maximum activation are shown.

Table 4: Acceptance Ratio of upward FlexRequests with relaxation of maximum activation per asset

	Range of energy of FlexRequests: [0-10%]		
	Proposed Approach	**Benchmark 1**	**Benchmark 2**
Acceptance Ratio of FlexRequests in terms of energy	39.96%	47.44%	28.97%
Acceptance Ratio (Positive responses to Flex Requests)	45.83%	54.17%	43.75%
	Range of energy of FlexRequests: [0-20%]		
	Proposed Approach	**Benchmark 1**	**Benchmark 2**
Acceptance Ratio of FlexRequests in terms of energy	10.14%	19.68%	8.19%
Acceptance Ratio (Positive responses to Flex Requests)	16.67%	25%	16.67%
	Range of energy of FlexRequests: [0-30%]		
	Proposed Approach	**Benchmark 1**	**Benchmark 2**
Acceptance Ratio of FlexRequests in terms of energy	9.89%	15.75%	7.81%
Acceptance Ratio	31.25%	35.42%	29.16%

(Positive responses to Flex Requests)			

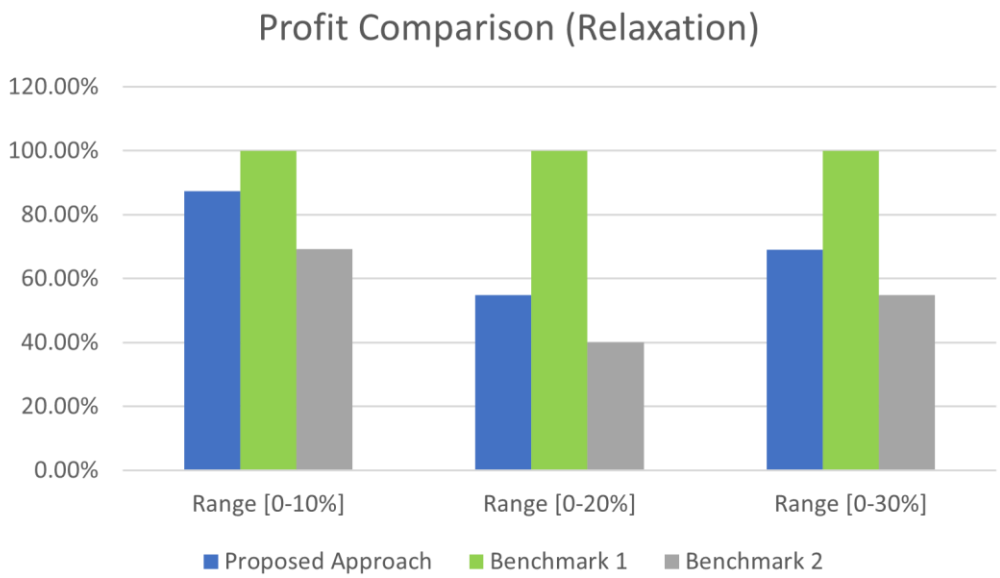

Figure 7: Comparison of profits of the proposed approach and the two benchmarks with a relaxation of maximum activation per asset

With the increase of the available flexibility, the difference in the performance of the three approaches is more evident. Shiftable assets can be more effectively used and prior information allows better scheduling of the portfolio.

It is interesting to note, that the increase to the limit of activation of adjustable assets leads to an increase of the use of shiftable assets. The deviation of a shiftable asset causes an increase of consumption for one or more MTUs. As only requests for upwards regulation are considered, these increments of consumption are counterbalanced by activations of adjustable assets. It should also be noted that the execution time of the proposed approach for the entire time horizon is within minutes, thus scalability of the solution is plausible and realistic.

2.6 Concluding remarks and summary of lessons learned

In FLEXGRID, UCS 4.1 focuses on the centralized management of distributed flexible assets and their representation in newly established flexibility markets. Within this context, a tool was developed to allow the aggregator to issue dispatch orders to its portfolio to respond to flexibility requests of the market. The independent aggregator is responsible for the flexibility of the assets and needs to coordinate with suppliers to establish the scheduled operating pattern of all assets within the portfolio and centrally coordinate the flexibility schedule when interacting with flexibility buyers and the flexibility market. End-users can register their assets to the aggregator's portfolio and create new revenue streams with their flexibility while ensuring that their preferences and constraints are respected, and they are treated in a fairly manner.

Upon the communication of the proposed approach and research results of UCS 4.1 with the relevant community, the following list has been formed, which summarizes the most important lessons learned and research and business insights.

Lesson learned	Research & Business insights
Coordination between suppliers and independent aggregators is needed in order to allow end-users to register their flexibility assets and participate in local flexibility markets. Scheduled operation and flexibility activation should be properly defined between involved market participants.	The sustainability of the aggregator, independent of the supplier representing only flexibility in the markets, needs to be examined. Flexibility can be offered independently based on a prior scheduled operation. Apart from flexibility after the DA market, different market floors for flexibility should be considered in parallel with existing energy markets (e.g. see UCS 2.3 in WP4 about stacked revenue maximization from participation in multiple markets).
A sustainable portfolio must consist of different types of end-users with different time offerings of flexibility (e.g. households, small business) to have diversity and available flexibility for multiple MTUs.	Available flexibility for multiple MTUs is critical for the aggregator to have a sustainable business model. Priority needs to be given for MTUs where it is more expected that flexibility will be needed.
Adjustable assets provide more usable flexibility to the aggregator as their operation does not affect multiple MTUs. Decrease of their operation to satisfy flexibility needed however may conflict with energy efficiency objectives and increase of operation cannot be currently sustained without the use of dynamic retail pricing.	Need for inclusion of generation and storage assets in the portfolio to validate the proposed approach for combinations of FlexRequests for both upwards and downwards regulation.
Shiftable assets can provide flexibility, without significantly affecting the comfort of the user. Shiftable assets with short operation schedules are more beneficial (less effect on other MTUs) but are more difficult to obtain scheduled operation and have direct control.	Flexibility needs other than those of TSOs are hard to come across and the value of flexibility from different market participants is in general not publicly available. Elaboration of FlexRequest concept is needed in order for FlexDemand market actors to implicitly declare their flexibility needs.
The sooner flexibility needs are known, the more effectively the aggregator can schedule and dispatch its assets, however the stochastic nature of needing flexibility must be taken into consideration for any proposed approach.	It is important to research, design and develop models and tools suitable for multiple market designs as the market is constantly evolving and adapting new technologies. Furthermore, potential interactions and combinations of market roles need to be taken into account. Several sources of uncertainty that may

	jeopardize the stability of the aggregator's business model should also be effectively modeled.

3 An aggregator manages its portfolio of many distributed flexibility assets dealing with many sources of uncertainty

3.1 Summary of FLEXGRID research results so far

In the previous chapter, we assumed that the aggregator is responsible for the management of the flexibility (flexible operation) of a portfolio of flexible assets. Flexibility is defined as the activated (controlled) deviation of the operation from the scheduled operating pattern. The scheduled operating pattern is determined by the supplier of the flexible assets and is submitted to the independent aggregator. In the current European market design, this refers to energy purchases and commitments in the Day-Ahead (DA) market, which correspond to the supply of the planned operation of the flexibility assets.

In this chapter, we consider an economic dispatch problem for an aggregator's portfolio, **where energy management decisions are made online and under uncertainty. We model multiple sources of uncertainty such as RES, wholesale electricity prices as well as the arrival times and energy needs of a set of flexible assets (i.e. Electric Vehicles).** The economic dispatch problem is formulated as a multi-agent Markov Decision Process. **The difficulties lie in the curse of dimensionality and in guaranteeing the satisfaction of constraints under uncertainty, which extends the research idea and respective problem formulation of the previous chapter.** A novel method, that combines duality theory and deep learning, is proposed to tackle these challenges. In particular, a Neural Network (NN) is trained to return the optimal dual variables of the economic dispatch problem. By training the NN on the dual problem instead of the primal, the number of output neurons is dramatically reduced, which enhances the performance and reliability of the NN. Finally, by treating the resulting dual variables as prices, each distributed agent (i.e. flexible asset) can self-schedule, which guarantees the satisfaction of its constraints. As a result, our simulations show that the proposed scheme performs reliably and efficiently.

Conclusively, the main differences of this chapter's system model and problem formulation (compared to chapter 2) are :

- Scheduled operating pattern of flexibility assets is not known beforehand and thus decision-making process is online.
- In Chapter 2, the aggregator represents only the flexibility of the assets of the portfolio (participation in flexibility market) and a supplier is responsible for the energy of the scheduled operating pattern. However, in this chapter, the aggregator is also responsible for the imbalances compared to the day-ahead energy schedule.

3.2 Problem Statement, related state-of-the-art and FLEXGRID research contributions

In modern power systems, there is an increasingly high penetration of small Distributed Energy Resources (DERs), such as rooftop solar panels, micro-generators and flexible

controllable loads, predominantly Electric Vehicles (EVs). Moreover, **many of these DERs exhibit high levels of uncertainty, in the sense that their constraints, costs and parameters are not deterministic**. The integration of DERs into electricity systems has motivated hierarchical market structures where groups of DERs interact with the system as a single (aggregated) entity. These aggregation schemes can take various forms, e.g. demand response or flexibility aggregators, virtual power plants, energy collectives, while the group-forming DERs may or may not reside at the same geographical location, depending on the use case and regulations. Such groups of DERs are also often called Energy Communities (ECs) [1], where the EC exchanges power with the system and an EC manager entity (or else aggregator) performs the energy management within the EC, i.e., coordinates the energy profiles of the community's DERs and decides on the power exchange with the main system.

Towards making dispatch decisions within an EC, intra-community economic dispatch problems have been the topic of several studies. EVs appear to be the focus of a significant portion of this literature, partly because they are considered to be the predominant source of end-use flexibility to be integrated in the medium and long-term future. The dispatch of the EVs (or, more generally, the DERs) of an EC can be realized via direct control, or via an intra-community market such as the B2C flexibility market proposed by FLEXGRID UCS 4.2 (see more details about this research thread in the next chapter below). Such B2C flexibility markets operated by an independent aggregator entity, have been proposed for various use cases, including prioritizing EV charging in a charging station [2], flexibility markets for providing local congestion management and voltage control services to DSOs [3] [4], and allocation of load curtailments by a demand response/flexibility aggregator [5].

Towards designing the mechanisms of such B2C flexibility markets, mechanism design (e.g. [6] [7]) and algorithmic game-theoretic approaches [8] have been proposed, while many studies (e.g. [9]) exploit duality to construct a price-based control scheme. However, in settings with high penetration of DERs, **the local economic dispatch problem involves the energy management (or else scheduling) of numerous small flexibility assets, and also needs to be solved in an online fashion and under uncertainty**.

Regarding these new challenges, a certain part of the literature has focused on **designing schemes for making efficient dispatch decisions under uncertainty**. There is certain differentiation among studies, regarding the way they treat uncertainty. The case of cost-efficient EV charging under the absence of forecasting tools using online optimization, is surveyed in [10]. Some studies, e.g. [11] [12] [13], **propose energy management schemes that solve a deterministic problem based on forecasts and then reassess the scheduling using a rolling horizon technique**. Some studies configure the energy management decisions with forecasting methods based on Machine Learning (ML). In [14], random forests are utilized to predict the uncertain parameters of flexible loads (EVs) before solving the economic dispatch problem, while [15] uses LSTM deep learning for a similar cause. However, in these approaches, **the uncertainty over forecasted parameters is not taken into account when making decisions**.

Another group of studies uses stochastic programming to sample realization scenarios for the uncertain system parameters. In [16], the authors propose a stochastic-robust approach for the decisions of a system with EVs. The applicability of the alternate direction

method of multipliers in stochastic B2C flexibility markets is investigated in [17], while in [18], stochastic programming is applied to an extended system that also includes resources across different energy carriers. In [19], a stochastic MILP formulation is proposed to assess the impact of RES and EV uncertainty on the energy management of a smart building, while in [20] and [21], the authors used stochastic dual dynamic programming to address RES uncertainty in dispatch problems. The authors in [22], configure the scenario-based method with information-gap theory to account for robustness. However, with multiple DERs, system parameters span over an exponentially large space, which means that **the relatively very small number of scenarios sampled by a stochastic programming method can fail to generalize reliably**.

A third family of studies leverages techniques from **Artificial Intelligence (AI) to account for decision-making under uncertainty**. The decision problem is modeled as a Markov Decision Process (MDP). In [23], a policy-rollout method is proposed to tackle the problem of home energy management under uncertain electricity prices. In [24], a policy-improvement method is proposed, which is warm-started by assuming a “good" policy learned by experience. In [25], a battery is controlled using approximate dynamic programming. **Dynamic programming is a standard approach for tackling MDPs; however, the curse of dimensionality prevents this family of methods from generalizing to problems with multiple agents (e.g., EVs), unless simplifying assumptions are made**.

Towards managing these issues, hybrid ideas have been proposed in the literature. The authors in [26] use dynamic programming to decide on the aggregated energy of an EV fleet, while an auction is used to allocate this energy among the EVs. A similar approach is taken in [27] for a system that also features RES generation. This technique is case-specific, since it builds on the assumption that the energy needs of EVs can be modeled simply by departure constraints and therefore aggregated to form a single-agent MDP. Another approach is to apply a standard Lagrangian decomposition to the economic dispatch problem of the aggregator, where each DER solves a MDP to decide on its own dispatch, using the iteratively updated Lagrange multipliers calculated by the aggregator. In such studies (e.g. [28]), **the aggregator (or else community manager) treats the DER responses as if they were deterministic, and all uncertainty is virtually delegated to the DERs. It should be noted that such a procedure is not provably convergent**.

A third approach is for the **aggregator to use a Machine Learning (ML) algorithm to learn optimizing the actions (dispatch decisions) directly, in line with the so-called learning to optimize framework** [29], which has also been applied to power systems recently (e.g. [30]). **This is in contrast to the studies that use ML only for forecasting system parameters, and then solve a deterministic optimization problem**. In [31], a deep reinforcement learning algorithm is proposed for making online dispatch decisions for EV charging stations. In [32], price-based control is realized via reinforcement learning, while a Neural Network (NN) is used as a function that maps prices to the agents’ response. In [33], ML algorithms are trained (using the system’s history) to make real time decisions on EV energy management. **An important drawback of these methods is that they cannot handle constraints explicitly, i.e., constraints are satisfied only in expectation** [34]. In [35], a penalty term is designed to teach a NN to also respect the local constraints of the agents. As analyzed in [36], the design of such a penalty term comes with various trade-offs (e.g., efficiency is sacrificed in order to

guarantee constraint satisfaction). The authors in [36], applied constrained policy optimization [37], which guarantees constraint satisfaction, in a setting involving an EV that is charging under dynamic electricity prices. However, **the method introduces high complexity and constraint satisfaction is relaxed in order to make the method faster. Moreover, the authors consider only one EV, so the curse of dimensionality that occurs in multi-agent MDPs is not addressed**.

Within the FLEXGRID's WP3 research work context, **we formulate an economic dispatch problem as a multi-agent MDP, which cannot be tackled by standard dynamic programming algorithms. Assuming a coordinating entity (i.e. independent aggregator), we apply a machine learning algorithm in the dual problem space and take advantage of the fact that its dimension (number of variables) is drastically smaller than the one of the primal problem. Moreover, the proposed method is able to guarantee constraint satisfaction.** Our contributions can be summarized as follows:

- We consider multiple agents and multiple sources of uncertainty, i.e., a number of EVs with deadlines, RES and inflexible demand, as well as uncertain real-time electricity prices for drawing energy from the main system. Conventional generation cost is also modeled and a power balance constraint couples the decisions of all agents.
- A Neural Network (NN) is trained to perform energy management decisions in real-time, upon receiving the information about the current system state. Instead of the primal problem, the dual problem is used to train the NN. With this technique, we reduce the NN's mean absolute error and enhance the NN's reliability.
- We propose an algorithm for energy management, through which the satisfaction of all the constraints is guaranteed. In particular, the distributed flexible loads (EVs) are allowed to self-schedule based on the dual variables provided by the NN.
- We perform energy management in a rolling horizon fashion, so that the NN can adapt its decisions based on new information about the system's state.
- The algorithm's performance is experimentally evaluated using a generic but realistic setup with convex generator models and EVs.
- Our results indicate that the proposed method achieves a near-optimal performance and significantly outperforms the conservative and offline constraint-satisfying benchmark.

3.3 System Model

We consider an economic dispatch problem in a setting with conventional generators, RES generation, inflexible demand and a set of EVs that ask for charging services upon arrival. In what follows, we model the operational characteristics of an aggregator for a certain time horizon T, where continuous time is divided into discrete timeslots of equal duration.

The aggregator[2] features a set $\boldsymbol{G}$ of power generators. The power output of a generator $\boldsymbol{j} \in \boldsymbol{G}$ in timeslot $\boldsymbol{t} \in \boldsymbol{T}$ is denoted by decision variable $\boldsymbol{g_{j,t}}$. Each generator is characterized by its lower and upper operational limits as follows:

[2] The terms "aggregator" and "energy community - EC" are used interchangeably in this chapter having a similar meaning.

$$g_j^{\min} \leq g_{j,t} \leq g_j^{\max}, \quad \forall j \in G, t \in T \qquad (3.1)$$

and also bears a cost function $C_j(g_{j,t})$ that maps its output to a certain monetary cost. $C_j(\cdot)$ is taken in this work to be a quadratic function:

$$C_j(g_{j,t}) = \mathrm{c}_j(g_{j,t})^2, \quad \forall j \in G, t \in T. \qquad (3.2)$$

The aggregator draws energy from the main electricity system at a time-varying per-unit price l_t. The vector of prices for all timeslots is denoted as $\mathrm{l} = \{\mathrm{l}_1, \mathrm{l}_2, \ldots, \mathrm{l}_{|T|}\}$. The aggregated energy drawn by the aggregator at t is denoted as $g_{0,t}$, and it is constrained by an upper bound K, i.e.:

$$\boldsymbol{g_{0,t}} \leq \mathbf{K}, \quad \forall \boldsymbol{t} \in \boldsymbol{T} \qquad (3.3)$$

The inflexible demand of the aggregator at t is denoted as $p_{infl,t}$, and the aggregator's RES generation as $g_{RES,t}$. The respective vectors containing the parameter values for all timeslots are denoted with a bold symbol, i.e. p_{infl},, = $p_{infl,1}$, $p_{infl,2}$, … $p_{infl,T}$,.

The aggregator also features a set of chargers for electric vehicles (EVs). The aggregator is responsible for satisfying a set of charging tasks $\boldsymbol{A}$, where a charging task $\boldsymbol{i} \in \boldsymbol{A}$ refers to allocating a certain amount of energy to an EV. Throughout this work, we refer to "EVs" and "tasks" interchangeably. Control variable $\boldsymbol{p_{i,t}}$ denotes the amount of power allocated to task $\boldsymbol{i}$ in timeslot $\boldsymbol{t}$. A power balance constraint, makes sure that the power generated equals the power consumed in every timeslot:

$$g_{0,t} + \mathrm{g}_{\mathrm{RES,t}} + \sum_{j \in G} g_{j,t} = \mathrm{p}_{\mathrm{infl,t}} + \sum_{i \in A} p_{i,t}, \qquad \forall t \in T \qquad (3.4)$$

A task $i \in A$ is characterized by a tuple $\Omega_i = \{a_i, b_i, d_i, p_i^{\min}, p_i^{\max}, E_i, \delta_i\}$, where a_i is the task's arrival time, b_i is the task's desired completion time, d_i is a completion deadline, $p_i^{\min}$ and $p_i^{\max}$ are the lower and upper bounds on the EV's power consumption, E_i is the total energy required for the task to be considered satisfied, and δ_i is a flexibility parameter that relates to the disutility that comes from delayed task satisfaction. Note that a task bears a departure time (deadline) d_i, upon which it must have received its required charging, but it is preferable to i to receive charging in earlier timeslots (preferably before b_i). Naturally, it is $b_i \leq d_i$. A bold symbol $\Omega_A = \{\Omega_i\}_{i \in A}$ denotes an instance of all task tuples. No energy can be allocated to a task, before the task's arrival time or after its completion deadline (i.e. after the EV departs):

$$\boldsymbol{p_{i,t}} = \mathbf{0}, \quad \forall \boldsymbol{t} \notin [\boldsymbol{a_i}, \boldsymbol{d_i}], \boldsymbol{i} \in \boldsymbol{A} \qquad (3.5)$$

while a task $i \in A$ can only be charged between the minimum and maximum charging rates of the respective EV:

$$\boldsymbol{p_i^{\min} \le p_{i,t} \le p_i^{\max}}, \quad \forall \boldsymbol{i} \in \boldsymbol{A}, \boldsymbol{t} \in [\boldsymbol{a_i}, \boldsymbol{d_i}]. \qquad (3.6)$$

The energy requirement of the task, has to be satisfied before the task's deadline:

$$\boldsymbol{\sum_{t\in[a_i,d_i]} p_{i,t} = E_i}, \quad \forall \boldsymbol{i} \in \boldsymbol{A} \qquad (3.7)$$

Finally, when the task charges at timeslots later than its desired completion time b_i, it suffers a cost (disutility) as follows:

$$\boldsymbol{U(p_{i,t}) = \frac{\delta_i^{t-b_i} p_{i,t}}{E_i}} \qquad (3.8)$$

where δ_i is a constant parameter. Intuitively, the term $\delta_i^{t-b_i}$ penalizes charging in timeslots later than the desired departure time b_i and favors earlier timeslots. Function $U(p_{i,t})$ can also be interpreted as the compensation that i requires from the aggregator, in order to shape its consumption profile.

Further constraints (e.g. power flows and voltage limits) could also be incorporated at this point. However, for simplicity of exposition, we assume that the necessary flow analysis has been conducted beforehand and that the limits posed by constraints (3.1), (3.3), and (3.6) already ensure that the physical grid operates within safe operational margins.

3.4 Problem Formulation

If all the information of the system (i.e., RES generation g_{RES}, inflexible demand $p_{RES,t}$, electricity prices l and tuples $\boldsymbol{\Omega}_A$ for the charging tasks of set A) was known beforehand, the cost minimization problem of the aggregator would be a deterministic, convex optimization problem, which reads as:

$$\min_{g_{0,t}, g_{j,t}, p_{i,t}} \left\{ \sum_{t\in T} \left(g_{0,t}\mathrm{l}_t + \sum_{j\in G} C_j(g_{j,t}) \right) + \sum_{i\in A} \sum_{t\in T} U(p_{i,t}) \right\} \qquad (3.9)$$

s.t. (3.1) – (3.8)

However, all the above-mentioned system parameters are not known beforehand (e.g. the EVs with their charging tasks arrive stochastically within the horizon without prior notice). **Therefore, the optimal aggregator's operation becomes a problem of decision making under uncertainty**. We assume that the aggregator has access to historical data or statistical information about the uncertain parameters, except for parameter δ_i, which is intrinsic to each user and the aggregator is only able to infer a relevant interval of δ_i[3]. Then, the problem structure is modeled as a Markov Decision Process (MDP) $\mathcal{M}$, defined as follows.

[3] In this paper, we treat the environment as stationary, but in Section 3.6.4 below, we also perform a sensitivity test for the proposed method. It should be noted though, that in cases of strongly non-stationary environments, it might be more suitable to adopt a reinforcement learning or a robust optimization approach depending on how critical the system is considered to be.

3.4.1 Formulation of Markov Decision Process $\mathcal{M}$

- **State:** $\left\{\tau, \boldsymbol{\Omega}_A(\tau), \left\{\sum_{t\in[a_i,\tau-1]} p_{i,t}\right\}_{i\in A}, \mathrm{g_{RES,\tau}}, \mathrm{p_{infl,\tau}}, \mathrm{l_\tau}\right\}$
  The setting's *State* consists of all relevant information available at a given system instance. *State* variable τ denotes the current operation timeslot. Variable $\Omega_A(\tau) = \{\Omega_i\}_{i\in A/\{\iota\}_{a_\iota>\tau}}$, denotes the tuples of tasks i that have arrived up to τ (i.e. $a_i \leq \tau$). Tasks that have not arrived yet are not included in the *State* information, since their tuples are unknown at τ. Variable $\sum_{t\in[a_i,\tau-1]} p_{i,t}$ is the power that has been allocated to task i up until timeslot $\tau-1$. Parameters $g_{RES,\tau}$, $p_{infl,\tau}$, l_τ refer to the current operational timeslot τ.

- **Actions:** $\{g_{0,\tau}, \{g_{j,\tau}\}_{j\in G}, \{p_{i,\tau}\}_{i\in A}\}$
  The action space is spanned over the possible values of decision variables $g_{0,\tau}$, $g_{j,\tau}$ and $p_{i,\tau}$.

- **Transition Functions:**
  - $\tau \rightarrow \tau + 1$
    After a decision on the power allocation for a certain timeslot τ is made, the system transitions to the next timeslot $\tau+1$.
  - At the next timeslot $\tau+1$, uncertain parameters mentioned above evolve stochastically[4].
  - $\left\{\sum_{t\in[a_i,\tau-1]} p_{i,t}\right\}_{i\in A} \rightarrow \left\{\sum_{t\in[a_i,\tau]} p_{i,t}\right\}_{i\in A}$
    At the next timeslot $\tau+1$, the new allocated energy for each task is calculated by adding the allocation decision $p_{i,\tau}$ of the last timeslot, to the the previously allocated energy $\sum_{t\in[a_i,\tau-1]} p_{i,t}$.

- **Cost:** $\mathrm{l}_\tau g_{0,\tau} + \sum_{j\in G} C_j\left(g_{j,\tau}\right) + \sum_{i\in A} U\left(p_{i,\tau}\right)$
  The cost of a certain *Action* at a certain *State* is defined as the sum of the energy procurement cost and the aggregated disutility.

The goal is to find a policy π, i.e., a mapping from each *State* to an *Action*, which minimizes the expected cost $J(\pi)$

$$\boldsymbol{J(\pi)} = \mathbb{E}_{\boldsymbol{\psi\sim\pi}}\left[\sum_{\boldsymbol{t\in T}}\left(\boldsymbol{g_{0,t}l_t} + \sum_{\boldsymbol{j\in G}} \boldsymbol{C_j}\left(\boldsymbol{g_{j,t}}\right)\right) + \sum_{\boldsymbol{i\in A}}\sum_{\boldsymbol{t\in T}} \boldsymbol{U}\left(\boldsymbol{p_{i,t}}\right)\right] \quad (3.10)$$

of the aggregator in the horizon T, where $\psi\sim\pi$ is the set of *State-Action* trajectories conditioned over policy π. Moreover, the policy must belong to the set F_π, which contains the policies that respect the constraints (3.1) – (3.8):

$$\boldsymbol{\pi}^* = \mathrm{argmin}_{\boldsymbol{\pi\in F_\pi}} \boldsymbol{J(\pi)} \quad (3.11)$$

[4] The transition function of the uncertain parameters is not necessarily known to the aggregator or even well-defined. It is assumed, though, that the aggregator has access to historical data of the uncertain parameters or, alternatively, is able to generate such data from estimated (possibly independent) transition functions of uncertain parameters.

The number of possible *States* and *Actions* in MDP $\mathcal{M}$ grows exponentially in the number of charging tasks, generators, and horizon timeslots, while $\mathcal{M}$ also features continuous *State* and *Action* variables. **Thus, problem (3.11) cannot be tackled by traditional dynamic programming algorithms.** Moreover, constraint (3.7) depends on the whole *State-Action* trajectory and not only on the current *Action*. This makes it very difficult to guarantee their satisfaction.

In particular, in standard MDP-solving frameworks, these kinds of constraints are incorporated in the *Cost* function of the MDP, multiplied by a penalty term ν, i.e. the MDP's *Cost* is extended with a term $\nu \sum_{i \in A} \left| \sum_{t \in [a_i, d_i]} p_{i,t} - E_i \right|$.

The issue is that if the value of ν is not large enough, constraint satisfaction is not guaranteed. On the other hand, choosing a very large value for ν forces the algorithm to pursue an overly conservative policy in order to make sure that constraints will be satisfied. Intuitively, in our context, an overly conservative policy would be to charge all EVs as fast as possible. This could lead to important efficiency loss as will be also shown in the simulations of section 3.6 below. In what follows we present a method that handles the intractability of the MDP $\mathcal{M}$, while constraint satisfaction is explicitly guaranteed.

3.5 Algorithmic solution for training the neural network and rolling horizon energy management

In this section, we present a novel method to tackle the MDP $\mathcal{M}$. We first present the high-level description of the algorithm. A very useful observation for MDP $\mathcal{M}$, is that its state variables can be conveniently separated in two special categories: the first category includes exogenous state variables $\Omega_A(\tau)$, $g_{RES,\tau}$, $p_{infl,\tau}$, l_τ, which have probabilistic transition functions but do not depend on the *Action* taken. The second category, includes τ and endogenous variables $\left\{\sum_{t \in [a_i, \tau-1]} p_{i,t}\right\}_{i \in A}$ (that depend on the *Action*), that both have deterministic transition functions. By exploiting these properties, the aggregator can run experiments for the setting, by sampling instances of $g_{RES}, p_{infl}, \mathrm{l}, \Omega_A$ from historical data or from known transition functions and, for each experiment, solve problem (3.9) as a deterministic problem to obtain an optimal solution (i.e. the optimal *Actions* in hindsight).

In a given experiment $\boldsymbol{k}$, the system transitions through a certain trajectory $\boldsymbol{\psi_k}$, of $|\boldsymbol{T}|$ *States* and respective optimal *Actions*. Therefore, after simulating an experiment $\boldsymbol{k}$, we can derive a set of $|\boldsymbol{T}|$ mappings, where a mapping $\boldsymbol{\mathcal{D}_S^{(k)}}$ expresses an association between *State* $\boldsymbol{S_t^{(k)}}$ (in which the system found itself) and the optimal *Action* (that the deterministic optimization method took in that *State*).

By running multiple offline experiments, the aggregator can generate multiple instances of $\boldsymbol{\mathcal{D}_S^{(k)}}$ (i.e. for a number of different *States*). These instances can be used to train a Neural Network (NN), towards learning to optimize the system. Thus, given an amount of Data $\boldsymbol{\mathcal{D}_S} = \{\boldsymbol{\mathcal{D}_S^{(k)}}\}_{\forall \boldsymbol{k}}$, the NN can be trained so as to provide a function that maps a *State* to an optimal *Action*. Therefore, in real-time operation, the aggregator can observe the system's current

State and feed it into the NN to obtain the NN's prediction of what would be the optimal *Action* in that *State*. The high-level procedure is illustrated in the figure below.

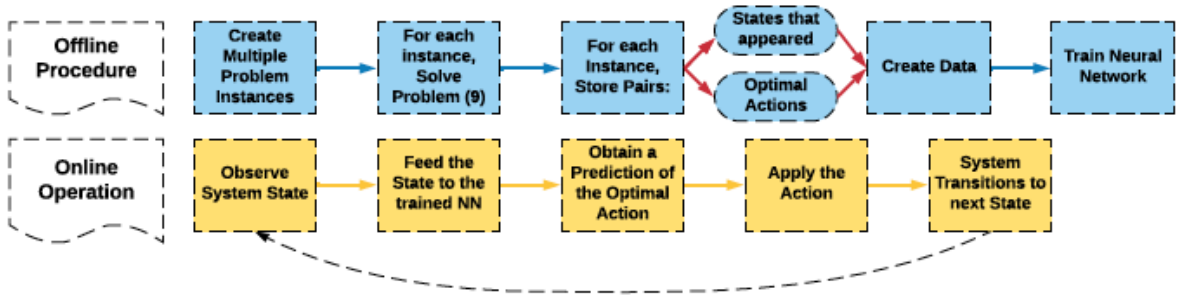

Figure 8: High-level procedures of data creation and online *Action* selection

There are two challenges with the proposed approach. First, there are $|\boldsymbol{T}|(\mathbf{1}+|\boldsymbol{G}|+|\boldsymbol{A}|)$ *Action* variables, which requires an equal number of output neurons. As a result, for only moderately large numbers of EVs in set $\boldsymbol{A}$, efficiently training the NN becomes quite challenging. Secondly, the NN does not guarantee constraint satisfaction, which means that if an EV's power allocation is determined by the NN output, it is not guaranteed that the EV will fulfill its charging requirements. In order to tackle these challenges, it is useful to consider the Lagrangian relaxation of problem (3.9). By relaxing constraint (3.4), the Lagrangian is written as:

$$\mathcal{L}(g_{0,t},g_{j,t},p_{i,t},\lambda_t) = \sum_{t\in T}\left(g_{0,t}\mathrm{l}_t+\sum_{j\in G}C_j(g_{j,t})\right)+\sum_{i\in A}\sum_{t\in T}U(p_{i,t}) - \sum_{t\in T}\left(g_{0,t}+\mathrm{g}_{\mathrm{RES,t}}+\sum_{j\in G}g_{j,t}-\mathrm{P}_{\mathrm{infl,t}}-\sum_{i\in A}p_{i,t}\right)\lambda_t$$

where λ_t is the dual variable corresponding to constraint (3.4), for timeslot t. An EV's optimal response to a set of mutipliers $\lambda=\{\lambda_1,\lambda_2,\ldots,\lambda_{|T|}\}$ is defined as:

$$Q_i=\min_{p_{i,t}}\left\{\sum_{t\in T}\left(U(p_{i,t})+\lambda_t p_{i,t}\right)\right\} \quad (3.12)$$

s.t. (3.5) – (3.8)

while the response of the generation side is defined as:

$$R=\min_{g_{j,t},g_{0,t}}\left\{\sum_{t\in T}\left(g_{0,t}\mathrm{l}_t+\sum_{j\in G}C_j(g_{j,t})\right) - \sum_{t\in T}\lambda_t\left(g_{0,t}+\sum_{j\in G}g_{j,t}\right)\right\} \quad (3.13)$$

s.t. (3.1) – (3.3)

The dual problem is defined as:

$$\max_{\lambda_t} \left\{ R + \sum_{i \in A} Q_i \right\}$$
$$\text{s.t. } \lambda_t \geq 0, \forall t \in T \qquad (3.14)$$

and since the primal problem (3.9) is a convex optimization problem, strong duality holds and the solution λ^* to problem (3.14) achieves optimality for problem (3.9).

Based on these observations, we can define the aggregator's *Action* in terms of dual variables $\boldsymbol{\lambda_t}$ instead of primal variables $\boldsymbol{p_{i,t}}, \boldsymbol{g_{j,t}}, \boldsymbol{g_{0,t}}$, which dramatically reduces the *Action* variables from $|\boldsymbol{T}|(\mathbf{1} + |\boldsymbol{G}| + |\boldsymbol{A}|)$ to only $|\boldsymbol{T}|$. In turn, this greatly facilitates the NN training. According to this formulation, a piece of data $\boldsymbol{\mathcal{D}_S^{(k)}}$ derived from experiment $\boldsymbol{k}$, is defined as a mapping from a system *State* $\boldsymbol{S_t^{(k)}}$ to a set of dual variables $\boldsymbol{\lambda^{(k)}}$:

$$\boldsymbol{\mathcal{D}_S^{(k)} \equiv S_t^{(k)} \rightarrow \lambda^{(k)}} \qquad (3.15)$$

The exact procedure of the data generation step is described in Algorithm 1 shown below:

Algorithm 1 Data Generation for training the Neural Network

1: Initialize $k = 1$
2: **While** $k <$ number of experiments
3: Sample $\mathbf{g}_{\text{RES}}^{(k)}$, $\mathbf{p}_{\text{infl}}^{(k)}$, $\mathbf{l}^{(k)}$, $\boldsymbol{\Omega}_A^{(k)}$
4: Solve problem (9)
5: Store optimal primal variables $\{g_{0,t}^{(k)}, g_{j,t}^{(k)}, p_{i,t}^{(k)}, \ \forall t \in T, i \in A\}$
6: Store optimal dual variables $\boldsymbol{\lambda}^{(k)}$
7: **for each** $\tau \in T$
8: Create *State*
$$S_\tau^{(k)} = \left\{ \tau, \boldsymbol{\Omega}_A^{(k)}(\tau), \left\{ \sum_{t \in [a_i, \tau]} p_{i,t}^{(k)} \right\}_{i \in A}, \mathrm{g}_{\text{RES},\tau}^{(k)}, \mathrm{p}_{\text{infl},\tau}^{(k)}, \mathrm{l}_\tau^{(k)} \right\}$$
9: Create one piece of Data $D_S^{(k)} \equiv S_\tau^{(k)} \rightarrow \boldsymbol{\lambda}^{(k)}$
10: **end for**
11: $k = k + 1$

Figure 9: Algorithm 1 about data generation for training the Neural Network

In real-time operation, the aggregator only decides the dual variables $\boldsymbol{\lambda_t}$ (by observing the NN output) and communicates them to the EV-task agents. Thus, instead of a fixed power allocation, each agent is given a set of multipliers (prices), which it can use to solve its local problem and make sure that its local constraints are satisfied. At timeslot $\boldsymbol{\tau}$ of

online operation, agent $\boldsymbol{i}$ has already received power $\widetilde{\boldsymbol{p}}_{\boldsymbol{i,t}}$ for timeslots $\boldsymbol{t} < \boldsymbol{\tau}$. Thus, $\boldsymbol{i}$'s local problem at $\boldsymbol{\tau}$, is defined as follows:

$$\begin{aligned} &\min_{p_{i,t}} \left\{ \sum_{t \in T} \left(U(p_{i,t}) + \lambda_t p_{i,t} \right) \right\} \\ &\text{s.t. (5)–(8)} \\ &\quad p_{i,t} = \tilde{p}_{i,t}, \ \ \forall t < \tau \end{aligned} \tag{3.16}$$

After the agents decide their charging power, they communicate it to the aggregator, and the latter can make sure that the power balance constraint is satisfied in the most cost-efficient way, by solving the following minimization problem:

$$\begin{aligned} &\min_{g_{j,t}, g_{0,t}} \left\{ \sum_{t \in T} \left(g_{0,t} \mathrm{l}_t + \sum_{j \in G} C_j(g_{j,t}) \right) \right\} \\ &\text{s.t. (1)–(3), (4)} \\ &\quad g_{j,t} = \tilde{g}_{j,t}, \ \ \forall t < \tau \\ &\quad g_{0,t} = \tilde{g}_{0,t}, \ \ \forall t < \tau \end{aligned} \tag{3.17}$$

where, again, $\tilde{g}_{j,t}$ and $\tilde{g}_{0,t}$ denote the decisions made in previous timeslots. The decisions for the current timeslot are implemented, and the procedure repeats for the next timeslot in a rolling horizon fashion. The exact procedure is described in Algorithm 2 shown below.

Algorithm 2 Rolling Horizon Energy Management Algorithm

1: Initialize $\tau = 1$
2: **While** $\tau \in T$
3: Observe system *State* $S_\tau^{(k)}$
4: Feed *State* $S_\tau^{(k)}$ to the NN
5: Obtain the NN output $\boldsymbol{\lambda}$
6: Solve problem (16) using $\boldsymbol{\lambda}$
7: Solve problem (17), where for constraints (4), $p_{i,t}$ is given by the solution of problem (16)
8: Implement solutions $p_{i,\tau}, g_{j,\tau}, g_{0,\tau}$ for the current timeslot
9: $\tau = \tau + 1$

Figure 10: Algorithm 2 about rolling horizon energy management

Although the space of possible states is exponentially large and the NN's training dataset cannot possibly explore it, we expect that the NN can learn the underlying structure of the optimal set of multipliers and provide a near-optimal output at any *State*. Moreover, by requesting that the NN provides the dual variables as output (instead of the primal variables $g_{0,t}$, $g_{j,t}$ and $p_{i,t}$) we reduce the number of output neurons from $|T|(1 + |G| + |A|)$ to only

$|T|$, which enhances the NN's performance. At the same time, the agents' decisions are made locally and, thus, constraint satisfaction is enforced.

3.5.1 Neural Network

The NN of Algorithm 2 is implemented as a Multi-Layer Perceptron (MLP), a class of feed-forward artificial neural networks. MLP is essentially a supervised learning algorithm that learns a non-linear function approximator (most common is stochastic gradient descent) for either classification or regression. An MLP trains on a dataset and learns a function $\mathrm{f}(\cdot) : \mathrm{R}^{\mathrm{m}} \rightarrow \mathrm{R}^{\mathrm{n}}$ where m represents the number of dimensions for input, which is equal to the number of *State* variables in our case, and n represents the number of dimensions for output, which in our case is equal to $|T|$, since we have one dual variable for each timeslot.

The model of each neuron in each layer of the network includes a set of weighted inputs that are summed and passed through a nonlinear differentiable activation function. The purpose of our network is to receive as input all the (known) *State* variables at a given operational timeslot $\boldsymbol{\tau}$ and return the output prices $\boldsymbol{\lambda}$, one for each timeslot of the horizon $\boldsymbol{T}$.

For our purposes, the standard MLP needs to be re-modeled so as to allow dynamic input size. Given an operational timeslot τ of Algorithm 2, the tuples Ω_i of EVs that have not arrived yet (i.e., the state variables Ω_i for the EVs with $a_i > \tau$) are not observable. Therefore, the input size is dependent on τ. However, neural networks are, by design, restricted to accept fixed length input. In order to tackle this issue, a masking layer was added before the input layer. The alternative would be to train $|T|$ different networks, one for every timeslot of the horizon. In the final NN, all *State* variables were included in the input layer and when at a given τ a number of *State* variables is unobservable, their input is fixed to a specific value. The above-mentioned masking layer is essentially an additional array that records whether a value is actually present for a given input, or whether it is missing and thus should be skipped during the processing of the data. This technique is called data masking.

3.6 Simulation setup and performance evaluation results

3.6.1 Simulation setup

A total of 50 EVs (charging tasks) were considered $A = \{1,2,\ldots,50\}$. Unless stated otherwise, the data were generated from 1000 offline simulations, and, for each simulation, we considered a horizon T of 24 timeslots, where the horizon represents a two-hour interval, divided into 24 timeslots of 5-minute duration each. A two-hour interval with continuously increasing inflexible demand and electricity prices was chosen for the main experiments, in order to represent a rush hour / peak demand time, where the need for demand response is more probable to arise.

For each task $\boldsymbol{i} \in \boldsymbol{A}$,

- The lower bound $p_i^{\min}$ on the EV's charging rate was set to zero.
- The upper bound $p_i^{\max}$ was picked randomly in the set $\{2,3,4,\ldots,12\}$(kW).
- The arrival time a_i was picked randomly from set $\{3,4,\ldots,9\}$ for the first 17 tasks, and from set $\{14,15,\ldots,20\}$ for the rest of the tasks.

- The desired departure time b_i was set to $a_i + 3$.
- The required energy E_i was set as $E_i = b_i p_i^{\max} \left(\text{kWh}\frac{5}{60}\right)$.
- The deadline d_i was set to $d_i = b_i + \xi$, where ξ was picked randomly from set $\{1,2,3,4\}$.
- Parameter δ_i was picked randomly from interval $[1,1.25]$.

Two local generators were modeled, $G = \{1,2\}$, with technical minimum points $g_1^{\min} = g_2^{\min} = 0(\text{kW})$ and capacity limits $g_1^{\max} = 0.5|A|\max_{i\in A}\{p_i^{\max}\}(\text{kW})$ and $g_2^{\max} = 1000(\text{kW})$. The respective cost parameters c_j were set as $c_1 = 0.003$ and $c_2 = 0.01$, so as to simulate a base generator and a more expensive generator for demand peaks.

The system's electricity price $\mathbf{l_t}$ at timeslot $\boldsymbol{t}$ was generated by a random normal distribution with average value $\boldsymbol{\mu(l_t)}$ and standard deviation $\boldsymbol{\sigma(l_t) = 0.03}$. The value of $\boldsymbol{\mu(l_1)}$ (for timeslot $\mathbf{1}$) was set as $\boldsymbol{\mu(l_1) = 0.4}\frac{\$}{\text{kW}\frac{\mathbf{5}}{\mathbf{60}}}$. For later timeslots, the average value of $\boldsymbol{\mu(l_t)}$ was assumed to follow a Markov chain, where $\boldsymbol{\mu(l_t) = l_{t-1} + 0.02}$.

Similarly, for the inflexible demand $p_{infl,t}$, the standard deviation was set to $\sigma(p_{infl,\tau}) = 4$ and the average value was modeled as a Markov chain, where $\mu(p_{infl,t}) = p_{infl,t-1} + 4$,. For the first timeslot, it was set $\mu(p_{infl,1}) = 100$. Finally, RES generation $g_{RES,t}$ was set as $\mu(g_{RES,t}) = g_{RES,t-1} + 2.5$, $\sigma(g_{RES,t}) = 2.5$, and $\mu(g_{RES,1}) = 30$ KW.

3.6.2 Neural Network Design

The implemented architecture consists of a 5-layer structure, in which the input layer and the output layer are interconnected with two intermediate non-linear hidden layers, while the masking layer precedes the input layer. Each layer is dense and fully connected; the first hidden layer contains 150 hidden units, and the second contains 100 hidden units. The activation function chosen is ReLu, for computational simplicity.

The Adam algorithm[5] was used as optimizer; a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments. Mean squared error was used as loss function, and mean absolute error as our evaluation metric. The network was trained for 100 epochs. A dataset of 24,000 samples was generated, since, based on Algorithm 1, a number of $|\boldsymbol{T}|$ pieces of data are created at each one of the 1000 experiments. The $\mathbf{92}\%$ of the data were used for training, while the rest $\mathbf{8}\%$ was reserved for testing. Finally, it should also be noted that the data were scaled before being processed.

3.6.3 Benchmarks to compare our proposed scheme

We considered two benchmarks with which we compared the proposed method. The first is the optimal-in-hindsight solution, where we assume that the aggregator has perfect

[5] https://machinelearningmastery.com/adam-optimization-from-scratch/

knowledge over all uncertain parameters for the horizon T and solves problem (3.9) deterministically in order to acquire the optimal solution[6]. Naturally, this benchmark serves as a theoretical upper bound on the performance of the proposed method and its solution cannot be obtained in practice.

The second benchmark is the conservative solution, where each arriving EV is charged at its maximum possible rate $\boldsymbol{p_i^{\max}}$ until its demand is fulfilled. This would be the solution provided by a traditional MDP-solving algorithm, when accommodated with a large penalty term $\boldsymbol{\nu}\left|\sum_{\boldsymbol{t}\in[\boldsymbol{a_i},\boldsymbol{d_i}]}\boldsymbol{p_{i,t}}-\boldsymbol{E_i}\right|$ in order to guarantee that constraints (3.7) will be respected. For this case study, this solution can also be obtained by setting:

$$p_{i,t}^{(\text{conservative})}=\begin{cases}p_i^{\max}, & \text{for } t\in[a_i,b_i]\\ 0, & \text{otherwise}\end{cases} \tag{3.18}$$

and solving online dispatch/scheduling problem (3.17), where in constraint (3.4), it is set $p_{i,t} = p_{i,t}^{conservative}$.

3.6.4 Performance evaluation results

In this subsection, we present the simulation results. For a particular instance of the setting, the figure below depicts the aggregated consumption of EVs $\sum_{i\in A} p_{i,t}$ as resulted by Algorithm 2 and the conservative benchmark, on top of the net demand which is the inflexible demand minus the RES generation for each timeslot. From the figure below, we can observe that Algorithm 2, in contrast to the conservative solution, opts for a peak shaving in timeslots 6-7, 18-21 and a valley-filling in the respective consequent timeslots.

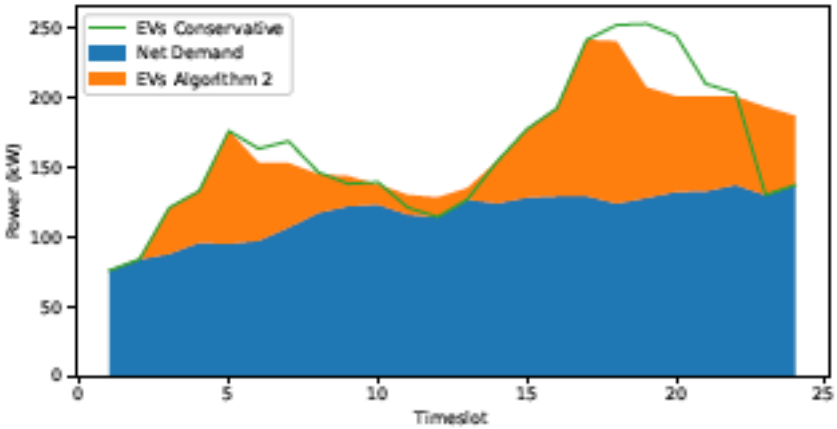

Figure 11: Electric Vehicles load on top of Energy Community's net demand for the proposed and the conservative case

Algorithm 2 and the two benchmarks were tested in a number $\boldsymbol{M}$ of different instances, where in each instance the testbed's parameters were sampled from the

[6] This deterministic optimization solution was followed by UCS 4.1 that is extensively described in chapter 2. It should be noted that UCS 4.1 solution will be integrated in FLEXGRID ATP (or else AFAT) to reach TRL 5. This chapter enhances the results of UCS 4.1 in TRL 3.

probability distributions described in section 3.6.1 above. The Social Cost $\boldsymbol{SC_{m,t}}$ for an instance $\boldsymbol{m}$ and timeslot $\boldsymbol{t}$ was calculated as:

$$SC_{m,t} = g_{0,t}\mathrm{l}_t + \sum_{j \in G} C_j(g_{j,t}) + \sum_{i \in A} U(p_{i,t})$$

The Social Cost for each timeslot was averaged out over all instances and the Cumulative Average Social Cost $\overline{SC_t}^{CuAv}$ is defined as:

$$\overline{\boldsymbol{SC_t}}^{\boldsymbol{CuAv}} = \frac{\sum_{\tau \in [0,t]} \sum \boldsymbol{g_{0,\tau}\mathrm{l}_\tau} + \sum_{j \in G} \boldsymbol{C_j(g_{j,\tau})} + \sum_{i \in A} \boldsymbol{U(p_{i,\tau})}}{\boldsymbol{M}} \quad (3.19)$$

The Cumulative Average Social Costs achieved by the proposed algorithm and the two benchmarks are presented in the figure below. The proposed approach on average achieves a 14.1% decrease in the aggregator's cost compared to the conservative benchmark, while it suffers a 11.4% higher cost than the optimal-in-hindsight solution.

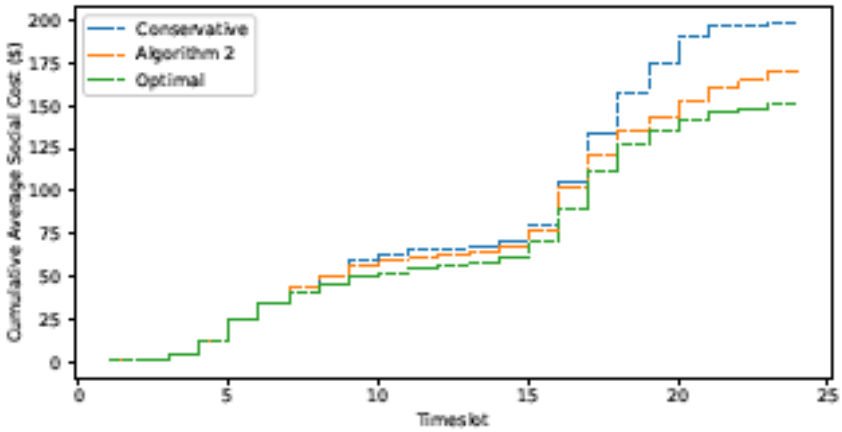

Figure 12: Average Social Cost achieved by the proposed algorithm compared to the two benchmarks

The proposed approach is able to achieve this performance, mainly for two reasons. Firstly, thanks to the small number of output neurons, the Neural Network achieves a very good performance towards tracking the optimal dual variables of the optimization problem (3.9). This is shown in the figure below, where the Probability Density Function (PDF) of the Mean Absolute Error (MAE) is depicted for both the train and the test dataset. Both PDFs follow a positively skewed normal distribution, the mode of which is around 0.02.

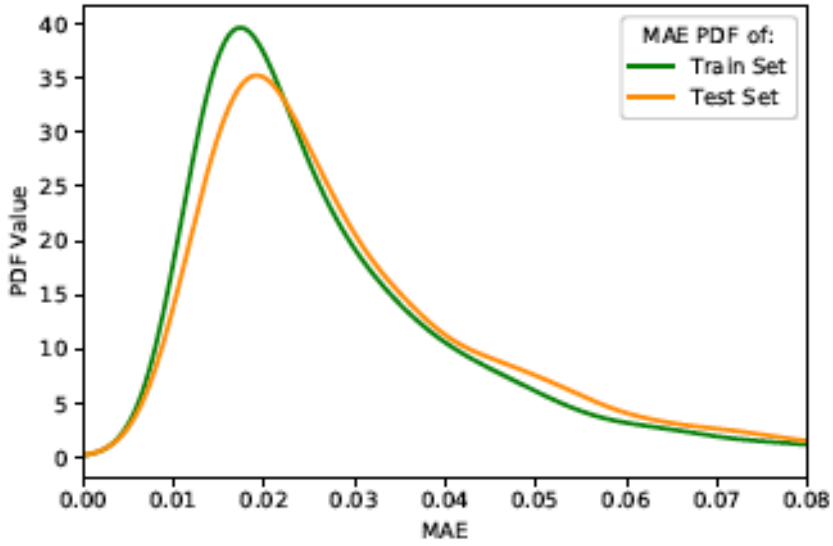

Figure 13: Probability Density Function of the Mean Absolute Error for the train and test dataset

Secondly, the rolling horizon approach of Algorithm 2 allows the NN to update the dual variables in an online fashion upon receiving new information on the system's *State*. Indeed, for later timeslots $\boldsymbol{\tau}$ (i.e., as time passes in the actual system operation), the NN's performance is improved, since it has accumulated more information about the problem's instance. This is quantified and presented in the figure below, where the MAE is depicted separately for each timeslot, i.e., the samples are grouped by the last known timeslot for which system parameters are observable.

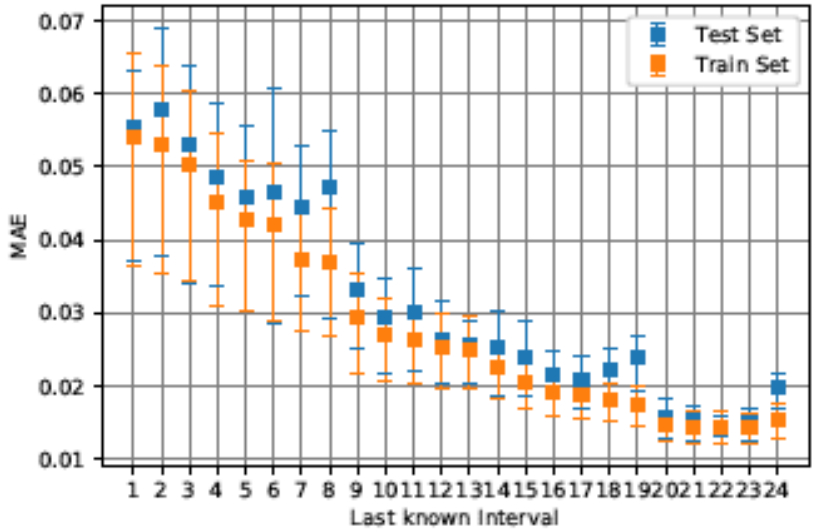

Figure 14: MAE box-plots (Q1, Q3) for the main train and test dataset

Moreover, as stated in line 8 of Algorithm 2, only the decisions for the current timeslot $\boldsymbol{\tau}$ are implemented in the system, and when the system transits to the next timeslot, the decisions are updated. In the figure below, the MAE for both timeslot $\boldsymbol{\tau}$ and the whole horizon $\boldsymbol{T}$ is depicted as a function of the training epochs. As can be observed, the MAE for the current timeslot $\boldsymbol{\tau}$ is even smaller compared to the MAE of the whole horizon. In practice,

since the procedure is repeated in a rolling horizon fashion and only the decision for $\boldsymbol{\tau}$ is implemented, it is more important that the NN achieves a good prediction of the optimal dual variable $\boldsymbol{\lambda_\tau}$ for the current timeslot $\boldsymbol{\tau}$ rather than for the whole horizon.

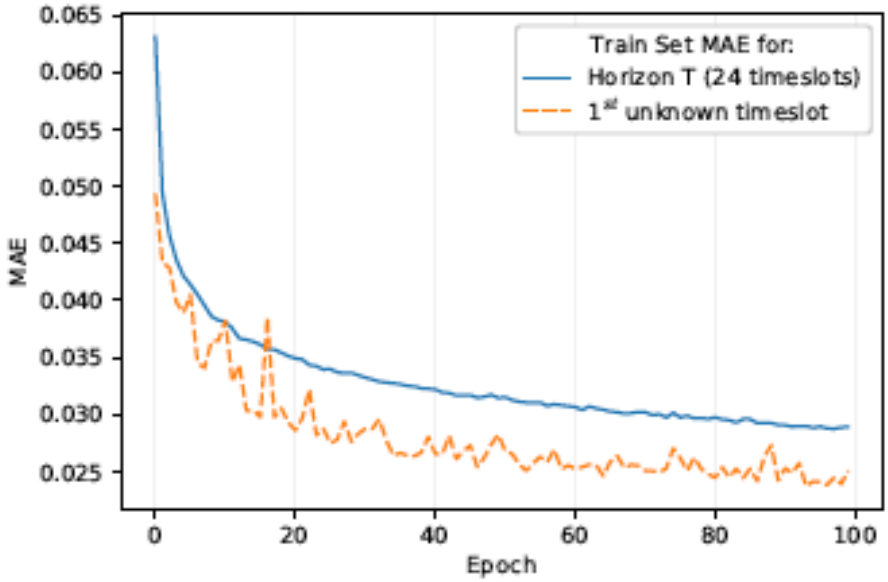

Figure 15: MAE obtained during 100-epoch period training, taking into account the predictions of the whole horizon T (blue), versus considering only the predictions of the first timeslot of each simulation (orange)

Next, we relax the assumption of a stationary environment (where historical datasets are representative of future realizations) with respect to the inflexible demand and electricity price. Specifically, we train the NN using $\boldsymbol{\sigma}(\mathbf{p}_{\boldsymbol{infl,t}})$ and $\boldsymbol{\sigma}(\mathbf{l}_{\boldsymbol{t}})$ as before, but we evaluate it using scenarios produced under (the different) standard deviations $\boldsymbol{f}\ \boldsymbol{x}\ \boldsymbol{\sigma}(\mathbf{p}_{\boldsymbol{infl,t}})$ and $\boldsymbol{f}\ \boldsymbol{x}\ \boldsymbol{\sigma}(\mathbf{l}_{\boldsymbol{t}})$respectively, where *f* is a factor by which the standard deviations are being altered. We evaluate the Competitive Ratio of Algorithm 2, defined as the ratio of the average optimal-in-hindsight objective value of problem (3.9) to the average social cost achieved by Algorithm 2, where the average is taken over the test scenarios. The figure below presents the results. The case of *f* = 1 represents accurate knowledge of the standard deviations. The algorithm's performance is not particularly sensitive to non-stationarity of electricity prices, but it is to the one of inflexible demand. More specifically, the algorithm's performance deteriorates as the error on the expected standard deviation of inflexible demand increases. When the actual $\boldsymbol{\sigma}(\mathbf{p}_{\boldsymbol{infl,t}})$ is as high as four times the value used for training, the algorithm's performance shows a significant deterioration. Nevertheless, for small errors, the algorithm's performance is not significantly affected.

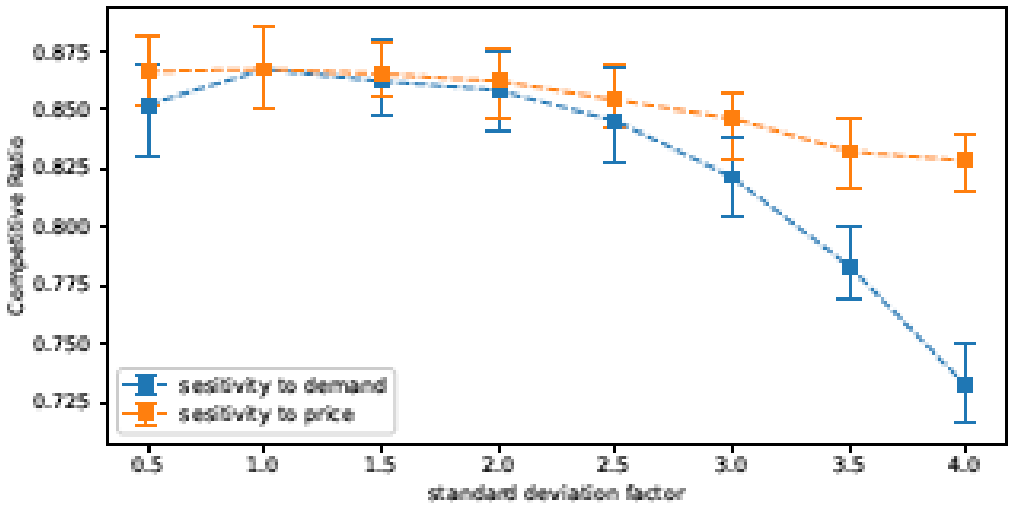

Figure 16: Sensitivity of Algorithm 2 to non-stationarity of the inflexible demand and electricity price

We further perform a number of tests to evaluate the performance of the proposed method under different cases, beyond the rush-hour case described in 3.6.1 above. Specifically, the average values of electricity prices, RES output and inflexible demand are no longer assumed to increase along T. Instead, the average value of $\mu(\mathrm{l}_t)$ was assumed to follow a Markov chain, where $\mu(\mathrm{l}_t) = \mathrm{l}_{t-1}$. An indicative set of 100 price trajectories, produced via random walks, is depicted in the figure below.

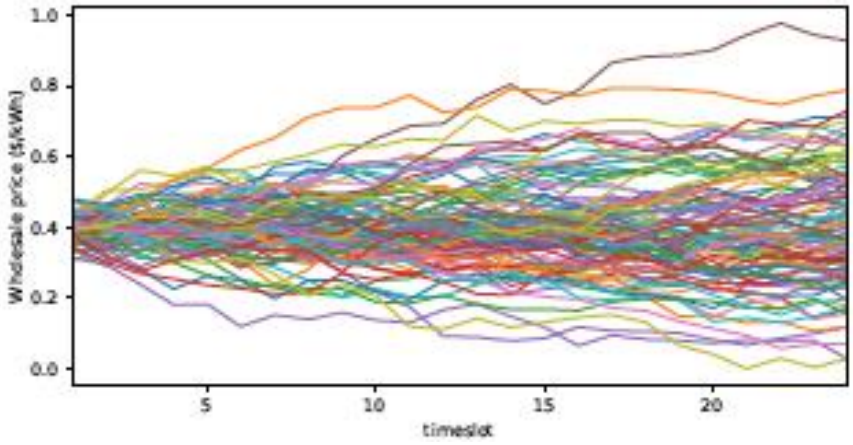

Figure 17: A set of random price trajectories produced by the assumed input pattern

For the inflexible demand and RES output, the average values were again modeled as Markov chains, but taken to increase for the first $|\boldsymbol{T}|/\mathbf{2}$ of timeslots, and decrease for the rest $|\boldsymbol{T}|/\mathbf{2}$, as in:

$$\mu(\mathrm{p_{infl,t}}) = \mathrm{p_{infl,t-1}} + \mathrm{sgn} \cdot 4$$
$$\mu(\mathrm{g_{RES,t}}) = \mathrm{g_{RES,t-1}} + \mathrm{sgn} \cdot 2.5$$
$$\mathrm{sgn} = \begin{cases} 1 & , \quad t < \frac{|T|}{2} \\ -1 & , \quad t \geq \frac{|T|}{2} \end{cases} \qquad (3.20) – (3.22)$$

Firstly, we evaluate the Cumulative Average Social Cost, as defined in (3.19) under this new setting. The results depicted in the first figure below validate the algorithm's performance in this setting as well. Next, we test the MAE of the NN under different horizon sizes $|T|$. The results, presented in the second figure below, verify the expectation that the MAE increases with longer horizons. Finally, in order to check if performance could be further improved by including more samples, we test the MAE achieved under different dataset sizes. As can be observed in the third figure below, the MAE is not significantly affected by performing more offline experiments on the setting.

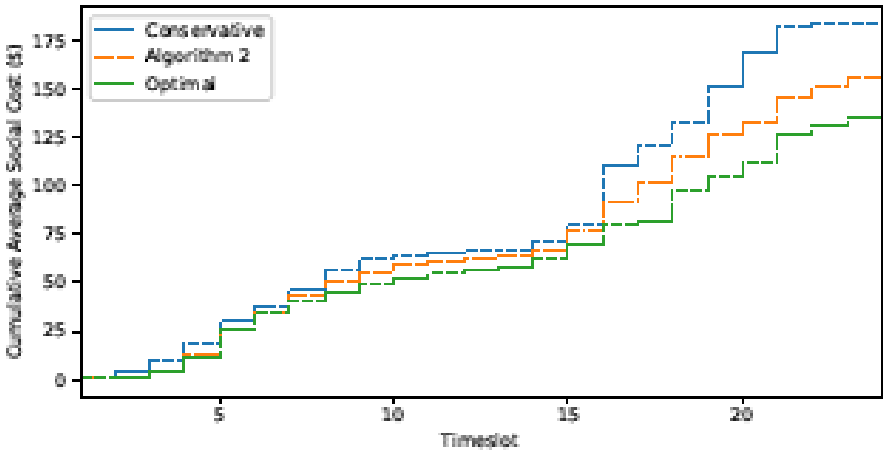

Figure 18: Average Social Cost comparison in the non rush-hour setting

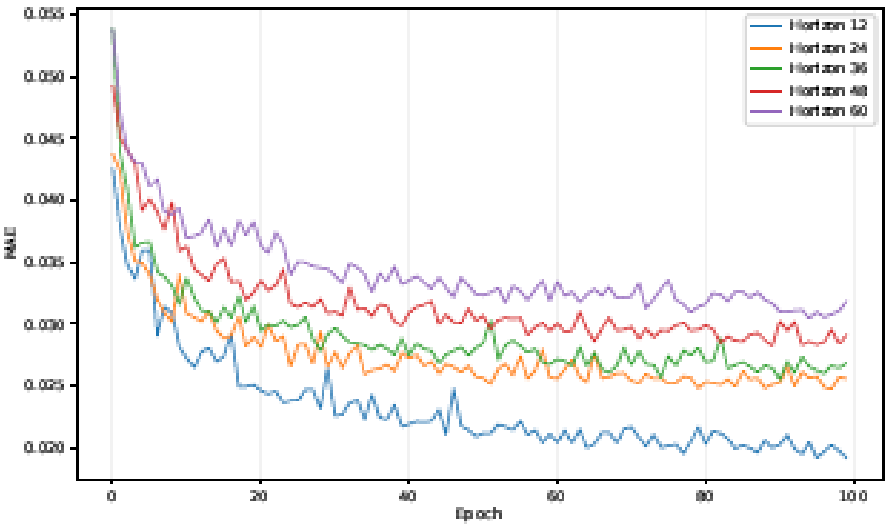

Figure 19: MAE under different time horizon sizes $|T|$

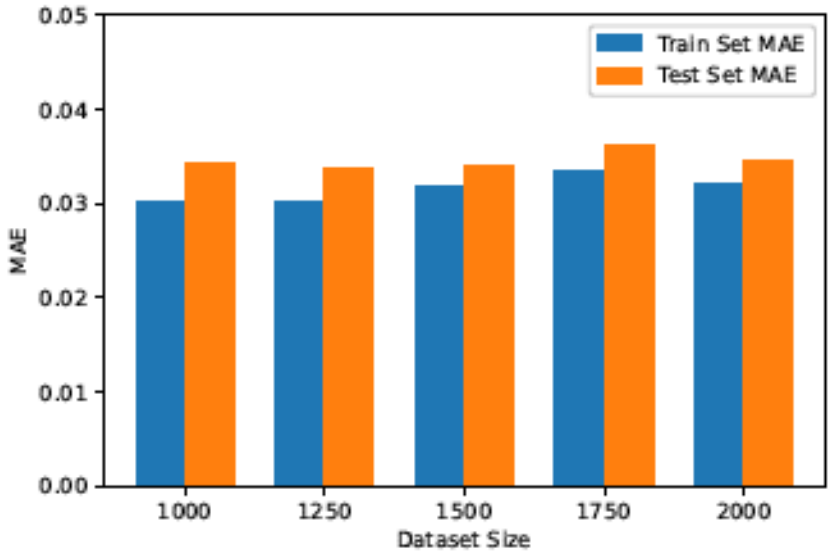

Figure 20: MAE of train and test set using different dataset sizes

3.7 Concluding remarks and summary of lessons learned

Conclusively, our work paves the way for considering the applicability of machine learning (ML) and Artificial Intelligence (AI) theory in the energy sector and more specifically in the smart grid research area. It also provides interesting research insights on how to more effectively model the decision-making process of several smart grid market stakeholders using the aggregator as an example. **The basic idea lies behind the concept of combining classic optimization theory with AI/ML in order to provide both more accurate (in terms of scientific excellence) and more practical (in terms of real-life business applicability and impact) solutions. With optimization theory, we make sure that we can achieve close-to-optimal results, while with AI/ML methods we can efficiently deal with the very complex processes, handling of huge amounts of data as well as dealing with many uncertainty factors, which are quite difficult to model in concise mathematical equations.**

After communicating FLEXGRID UCS 4.1 scientific results to both academic and industrial communities, we have come up with a short list of lessons learned that could be further investigated in future R&I initiatives. The table below summarizes research and business-related insights for each one of the lessons learned.

Lesson learned	Research & Business insights
AI/ML-based decision making for certain smart grid applications can take place in the dual problem space instead of the primal one. Thus, a rather complex problem can be transformed into a simpler one before a Neural Network is trained.	The proposed FLEXGRID mathematical and algorithmic methodology can be replicable in several other smart grid applications (OPF-based problems, optimal bidding/ scheduling/ planning) exploited by different market stakeholders (e.g. System Operators, ESPs, etc.)
The use of optimization theory can provide theoretically optimal results, but these may not be achieved in real-life business.	More research is needed on novel methodologies that are able to combine classic optimization theory with AI/ML-based

The use of AI/ML-based decision making may be highly inefficient and even not applicable in some business cases.	decision-making solutions in the energy sector in order to strive a good trade-off between scientific excellence and real-life business impact.
Optimal dual variables do not generally adhere to some "easy-to-learn" underlying data structure by a ML/AI algorithm. This means that the proposed FLEXGRID methodology could be a promising solution when considering other types of FlexAssets or even when dealing with other similar dual problem spaces.	Need for further validation of the proposed method when considering more types of FlexAssets at the same time (i.e. other than EVs, like curtailable/shiftable, battery storage, boilers, heat pumps, etc.), too. There is also need to consider more sources of uncertainty and test/ validate how these can further affect the performance evaluation results.
Results show case that the proposed methodology is not so sensitive to sampling errors. For instance, we trained the NN with one dataset and tested it with another dataset by deliberately inducing a noise pattern. Our solution performs well in the presence of unexpected outliers.	More validation tests are needed by utilizing real-life datasets from real-life FlexAssets and end users that may induce different types of outliers and noise patterns. Higher-TRL innovation activities are needed towards this direction.
The proposed methodology does not find the optimal solution, but it is scalable and very practical. Thus, there is no need to have the most accurate end user modeling, but just know some basic end user and FlexAsset parameters (maybe via a user-friendly mobile app) in order to be able to make good self-dispatch decisions.	Real-life pilot tests are needed at a higher TRL. Need to define what level of accuracy is needed in the end user/FlexAsset modelling and inter-relate with end user engagement requirements and corresponding key performance indicators (KPIs).
It is not easy for the aggregator to know the FlexAsset's model and the end user's utility model, because historical and statistical datasets are not generally available due to many legal, regulatory and technical restrictions. This increases the risk that the proposed solution may be not applicable in real-life business in the future.	Need for multi-disciplinary research by including social and behavioral sciences, legal and regulatory framework, etc. End users should be able to trust 3rd parties that they use their personal data only for mutually agreed purposes and be fairly compensated for the data that they provide. Efficient end user engagement and acceptance strategies should be followed towards this direction.

4 An aggregator operates an ad-hoc B2C flexibility market with its end energy prosumers by employing advanced pricing models and auction-based mechanisms

This chapter deals with the research problem of FLEXGRID UCS 4.2, in which a novel B2C flexibility market architecture is proposed. In this B2C flexibility market operated by an aggregator company, various types of small-scale Distributed Flexibility Assets (DFAs) may compete with each other in order for the required aggregated flexibility to be gathered by the aggregator (i.e. FlexSupplier) at the least possible cost.

4.1 Summary of FLEXGRID UCS 4.2 research results so far

In FLEXGRID UCS 4.2, we draw on concepts of mechanism design theory in order to define an iterative, auction-based mechanism, consisting of an *allocation rule* and a *payment rule*. The allocation rule refers to the way that the aggregator decides upon how much consumption reduction/increase will be allocated to each end user (i.e. energy prosumer) according to the feedback obtained through the auction process. The payment rule refers to the way the aggregator decides upon the reward of each user for his/her allocation, provided that the end user makes the corresponding contribution. Through the auction procedure, the aggregator exchanges messages with the end users in the form of queries. A query in our case is a price signal communicated from the aggregator to the end user, to which the end user responds with his/her preferred action (e.g. consumption reduction) according to this signal.

In the previous FLEXGRID D3.2, we **proposed advanced retail market mechanisms (ARMM) that can be used by an aggregator in order to operate a novel B2C flexibility market architecture**. A main research novelty of our work is that we consider the case in which an end user may respond untruthfully if he/she finds that to be in his/her interest.

The business value that FLEXGRID platform introduces for the aggregator user is that s/he will be able to run various “what-if” simulation scenarios (offline operation) in order to determine better ways (via retail pricing schemes) to operate a novel B2C flexibility market. In other words, **the aggregator will run a retail pricing algorithm to test and evaluate the impact that new FlexContracts (with its end users) would have on several KPIs** such as: i) aggregator’s revenues, ii) aggregated end users’ welfare, iii) quantity of flexibility offered to the system, iv) individual end user’s welfare. As a result, **the aggregator user will be able to intelligently identify how it can recommend a new (more beneficial) FlexContract to a set of end energy prosumers**. This novel FLEXGRID service is expected to help the aggregator to realize deep relationship with its customer portfolio and thus make it more competitive in the future retail/B2C flexibility markets.

In this deliverable, we elaborate on the UCS 4.2 research work in order to deal in more depth with algorithmic complexity and scalability problem. Our research findings from FLEXGRID D3.2 indicate the need to deal with the scalability problem, which becomes very difficult to solve when we consider a large number of FlexRequests published by the FLEXGRID ATP, a large portfolio of end users (i.e. at a scale of several hundreds of thousands of end users or even millions[7]), more complex (and thus more realistic) FlexAsset models and more stringent real-time constraints imposed by the emerging B2C/B2B flexibility markets.

In order to cope with these research challenges, our work within M19-M26 was focused on combining the existing work on B2C flexibility market operation (cf. D3.2, chapter 4) with an optimal cloud resource allocation algorithm. The proposed cloud resource allocation algorithm is able to service multiple FlexRequests (e.g. in multiple distribution networks), and minimize the cost of computational resources, while respecting the execution time constraints of each FlexRequest. This will motivate towards a cost-efficient and competitive B2C flexibility market as a service offering by the aggregator.

4.2 Problem Statement, related state-of-the-art and FLEXGRID research contributions

Towards facilitating the transition to a carbon-free electricity system, government policies introduce incentives for bottom-up investments in Renewable Energy Sources (RES). As a result, RES facilities are being installed in various locations of the medium or low-voltage distribution network. While these developments accelerate the penetration of RES, they do, however, create new challenges for power system operators. In particular, voltage and congestion issues become significant, as they can occur dynamically and close to real-time operation due to the volatile nature of RES output.

In order to avoid resorting to undesirable RES/load curtailments and costly grid reinforcement[8], Distribution System Operators (DSOs) can manage their networks and resolve voltage stability and congestion issues by drawing on the flexibility of small distributed resources located in the network, such as energy storage systems, micro-generation facilities and flexible loads, e.g., Heating Ventilation and Air Conditioning (HVAC), Electric Vehicles (EV), etc. We refer to these resources as *flexibility assets*, while the facility where they are located (building, charging station, etc.) is referred to as a flexible facility or simply *facility*.

Novel smart grid architectures such as the ones proposed by H2020 FLEXGRID project[9] prove market frameworks for flexibility activation in such contexts. The relevant marketplaces are commonly referred to as “flexibility markets”. A *flexibility market* is the marketplace where a DSO dynamically procures Demand Response (DR) services from assets

[7] Note that each end user may have several flexible electric appliances (FlexAssets), so the number of participating entities increases even more.
[8] This is the Business-As-Usual approach for DSOs today in the EU area.
[9] See more details about the various types of Distribution Level Flexibility Markets (x-DLFM) in FLEXGRID D2.2 (chapter 2) and D5.1 (chapter 2) here: https://flexgrid-project.eu/deliverables.html

located in different nodes of its network. In such markets, DR services are offered by flexible facility manager entities, who are responsible for aggregating, offering, and activating the facility's flexibility via DR actions. These DR actions are implemented by Energy Management Systems (EMS) that use Information-and-Communication Technology (ICT) to monitor and control the energy consumption of the flexible facility. Examples include Building EMS, EMS that monitor and control the charging power of electric vehicles in an EV charging station, and more.

The scalability properties of flexibility market-clearing algorithms constitute a critical issue towards bringing such solutions to real-life implementations. Moreover, each facility bears certain costs for performing DR actions, relating to the compensation (or energy bill discounts) that it should offer to its end users in order to modify the assets' energy consumption profile. The objective of the DSO is to satisfy the system's constraints in the most cost-efficient way, i.e., by drawing on the least expensive facility DR services. In addition, one needs to consider the minimization of the total system cost, including the DR procurement cost and the operational cost of the cloud services necessary to perform the various calculations required for the overall operation of the DR flexibility market.

4.2.1 Related works from the international literature

Cloud computing applications for the smart grid architecture have mainly focused on three areas, namely, energy management, information management, and security [39] [40]. However, the need to support flexibility markets that consider physical network constraints of the distribution network through Alternate Current Optimal Power Flow (AC-OPF) formulations, has recently emerged [41]. The computational complexity, the robustness [42] and the scalability [43] of these solutions pose critical demands, requiring the efficient allocation of DR flexibility markets' computational tasks to computational resources so that the delay and processing requirements that these architectures need are guaranteed in an economically efficient manner. Moreover, as stated in many recent survey works, such as [44] and [45], traditional cloud computing architectures can hardly meet the requirements of large-scale real-time data processing in DR applications. Therefore, novel cloud-fog-edge computing architectures have been recently proposed, in which computation tasks can be decomposed and be allocated to edge/fog nodes and clouds through more effective task allocation strategies to strike an optimal trade-off between various requirements, such as computational complexity, scalability, time-related constraints and total operating costs.

There are several works in the recent literature that propose a cloud-fog-edge architecture for dealing with DR operation. Ref [46] presents a pioneering work in the exploitation of an Edge-Cloud architecture towards efficient DR in buildings. Additionally, in [47], the authors analyzed communication performance as a major requirement in cloud-based DR. More specifically, a cost-effectiveness analysis confirms that achieving higher performance incurs a higher communication cost. However, neither of the aforementioned works have dealt with the issue of adapting the allocation of DR computation tasks to computing resources. Consequently, there are no overall DR performance (delay and scalability) guarantees, while the satisfaction of the physical constraints of the power distribution network, and the consequent computational load it entails, are not considered.

The work in [48] presents an important effort on the use of clouds towards real time DR services, which is often referred to as Emergency Demand Response. The efficiency of the proposed solution is testified through performance evaluation results. In the same direction, [49] exploits clouds towards the real time management of smart grids. However, the former study does not consider the underlying distribution network, while the latter does not consider DR services.

Furthermore, [50] proposes an integration between smart grid and cloud (noted as Internet of Energy) by proposing a smart gateway that bridges the fog domain and the cloud. It is introduced for scheduling devices/appliances by creating a priority queue that can perform demand side management dynamically. However, this paper only presents a communication architecture and does not model the algorithmic problems of resource allocation and flexibility market-clearing.

The work in [51] proposes a cloud-edge cooperative control model and strategy for the price-based DR of large-scale air conditioners, while it is compared with a classic single cloud architecture model. The results show reduction of the grid's critical peak and elimination of the peak rebound. However, the computation tasks are statically allocated to the cloud or the edge, while our work uses a diverse set of DR assets, which incurs the need for dynamic allocation of computing resources.

The authors in [52] propose a three-tier edge-cloud collaborative residential energy management architecture to alleviate fluctuations in demand, while reducing latency and improving processing performance. To this end, a two-level energy management mechanism was determined. The first stage models the interaction between real-time pricing and energy demand, while the second implements energy scheduling between the cloud tier, access tier, and infrastructure tier. [53] also proposes a similar 3-tier cloud-fog architecture that improves the response delay and uses a linearized AC-OPF model that finds the optimal solution. Edge computing resources are designed to generate Bender's cuts, and the cloud is designated as the coordinator of the whole process. Moreover, a few recent works, such as [54] and [55], deal with a distribution-level energy trading problem. [54] presents an energy trading management system, where the edge node acts as a retail energy market server providing energy services to the end-users. The architecture includes home gateways, local fog nodes and cloud server. The proposed edge/cloud model is compared to a classical single cloud-based one, showcasing its superiority with respect to network load and delay reduction. However, these works do not deal with optimal and dynamic allocation of computing resources and thus do not guarantee scalability and stringent delay constraints of the market clearing process.

A few more works, namely [56] and [57], consider cloud-edge architecture to deal with electric vehicle (EV) fleet management problem. In [56], cooperation among cloud and edge devices is realized to make intelligent decisions related to EVs' charging and discharging in addition to achieving the expected demand-supply balance, without accounting for distribution network constraints. [57] transforms a traditional large-scale V2G problem into several sub-problems, which are small enough to optimize. Network constraints are also taken into account. In our work, we model diverse DR assets and not only EVs. We also

propose an optimal solution for the orchestration of the heterogeneous cloud, fog and edge computing resources.

Finally, the authors in [58] proposed the energy management as a service concept, which is implemented over a fog infrastructure. Scalability, adaptability, delay constraints and cloud cost minimization are some of the requirements that are extensively discussed. However, this is rather a high-level analysis, which means that there is neither a mathematical problem formulation nor a proposed algorithm included. In contrast, **our work co-optimizes the cost of DR procurement and cloud resources by developing a solid mathematical model and algorithmic solution to realize the novel Demand Response Operation as a Service (DROaaS) business model**.

4.2.2 FLEXGRID research contributions

Motivated by the above-mentioned research challenges and related works, we propose, for the first time, an innovative business-to-business cloud service, noted as DR Operation as a Service (DROaaS), which facilitates aggregators, through the use of a DR-oriented dynamic cloud resource allocation framework. By exploiting intrinsic attributes of DR models to optimize cloud-based execution, the proposed framework provides the aggregator with a flexibility allocation algorithm that is:

- scalable in terms of number of assets and distribution network locations,
- dynamic and able to make fast, real-time decisions, and
- optimal (cost-efficient) in terms of minimizing the total system's cost (i.e., both DR procurement cost and cloud-related operational expenditures).

The major contributions of FLEXGRID UCS 4.2 are summarized as follows:

- A decomposition algorithm is presented, through which the flexibility market clearing problem is parallelized so that it becomes amenable to distributed (cloud) computation.
- An integrated framework is developed that facilitates the realization of the Demand Response Operation as a Service (DROaaS). This service calculates the optimal DR actions, while optimizing the cost of computational resources towards the proliferation of cost-competitive DR services.
- The use of computational resources in the proposed multi-technology DR architecture is optimized through an integer linear programming algorithm. A heuristic algorithm is also presented, that achieves near-optimal performance with negligible computational time.
- An extensive evaluation is performed, under a diverse set of end user devices and models. The evaluation results demonstrate the scalability, low delay and cost-competitiveness of the proposed architecture.

4.3 Proposed DROaaS architecture

The proposed DR architecture assumes a computing and networking (COMNET) infrastructure that interacts with the EMS and supports the operation of the DR mechanisms. The COMNET infrastructure combines heterogeneous resources from the edge/fog layers to

bring adequate resources close to flexibility assets, and from multiple clouds (federated operation).

Computing resources can range from generic ones, to specialized computing devices (FPGA, GPU), to micro Data Centers (DCs) and larger Data Centers (DCs), deployed in urban (office and residential buildings) and rural areas (e.g., alongside farms of wind turbines and solar panels), some closer to the edge and some deeper in the cloud forming the edge-fog-cloud hierarchy. Moreover, these resources may belong to different administrative authorities (providers) thus forming a hierarchy of privately owned and public computational resources. Moving from the lower layers of the hierarchy to the higher ones, the provided capacity, scalability and resiliency increase, but so does the delay. Edge resources can perform light computations and filtering functions, while complex computations have to be offloaded to the higher layers, i.e., deeper in the cloud. Such approaches are currently being adopted in other time-critical applications, e.g. closed-circuit television cameras fitted with artificial intelligence capabilities for facial recognition technology. We regard that the same approach is relevant for the smart energy field. More specifically, we consider as edge resources the resources that operate within and/or close to each facility featuring low network delay, but low computational capacity. We also note as fog resources the resources located on the local DSO data center (i.e. dedicated servers, which are available to compute purpose-specific applications). Finally, we note as cloud resources the ones located at large data centers and are typically owned by large cloud service providers (e.g., Amazon, Google, etc). These usually have larger network delay but higher capacity.

The networking infrastructure includes various networking mechanisms using different wired (optical) and wireless (e.g. 4G/5G) technologies to provide the required interconnection of the computing resources over private and public network infrastructures. These multi-domain and multi-technology network paths are controlled and managed by the telecom operators based on Software Defined Networking (SDN) principles. Hence, we abstract the communication paths between the resources in the same or different layers as virtual links with specific latency and capacity. These values depend on the networking locality of the resources, with those in proximity resulting in lower latency than those that are far apart.

Each facility's Energy Management System (EMS) infrastructure contains sensors, actuators and/or smart plugs, together with appropriate interfaces through which end users are able to set their preferences regarding the use of their flexibility assets for providing DR. By drawing on the EMS monitoring and control capabilities, the flexible facility can offer DR services to the aggregator. In turn, the aggregator needs to decide the optimal configuration of DR-services (e.g., which facilities should activate their flexibility and by how much). This optimization can be mathematically decomposed into smaller subproblems (computational tasks), which can be performed in a cost-effective and time-critical manner by drawing on the COMNET infrastructure. Thus, a business model is enabled, where an aggregator can act as an intermediary entity and thus offer a DR operation as a service (DROaaS) to multiple systems of DSOs and flexibility asset owners/end users.

The figure below depicts the proposed DR Operation as a Service (DROaaS) that can orchestrate the decomposed instances of the market clearing algorithm over the available COMNET infrastructure. The main components that form the proposed DROaaS architecture are:

- An EMS per facility that exploits ICT technology to:
  - monitor and control the flexibility assets of that facility,
  - allow end-users to declare their electricity consumption preferences through a user interface, and
  - communicate the facility's DR capabilities and receive dispatch orders.
- The Service Orchestrator provides the necessary interface between the DROaaS platform (i.e. residing at the aggregator side) and the facilities and DSOs. It receives, through its interface, the requirements and specifications of the DR market clearing problem (see more in sections 4.4.1-4.4.2).
- The Resource Orchestrator decomposes the DR market clearing problem (i.e. B2C flexibility market) into smaller subproblems (see more in section 4.4.3), and assigns the subtasks to the most appropriate edge, fog, or cloud computational resources (see more in section 4.5).
- The Infrastructure Manager handles the interaction with the local orchestrators at the various computational resources, models their capabilities and enables their monitoring.

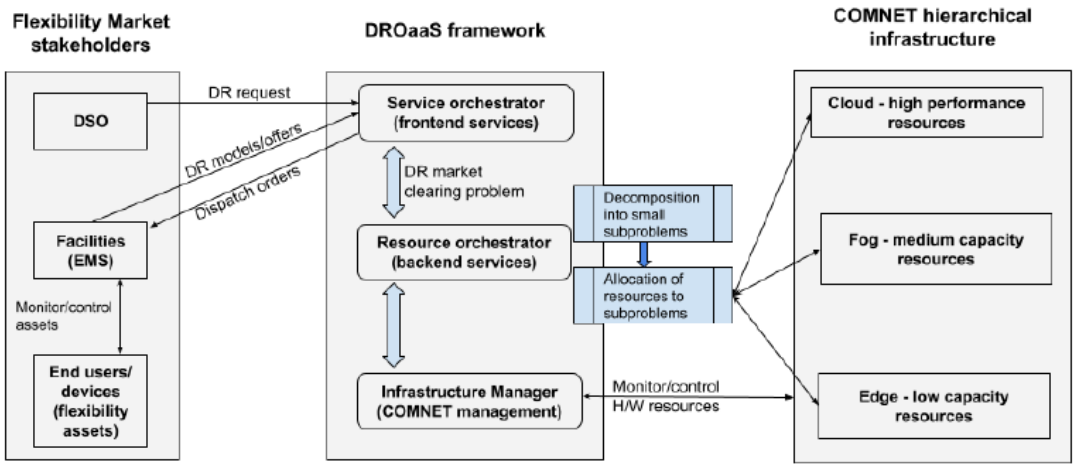

Figure 21: The proposed Demand Response Operation as a Service (DROaaS) architecture

In the following Section, we elaborate on the problem definition and decomposition, while in section 4.5, we present the algorithms for the allocation of the decomposed subproblems to computational resources.

4.4 Problem Formulation

4.4.1 System Model

A flexibility market consists of a set $N = \{0,1,2,\dots,|N|\}$ of flexible facilities (e.g., buildings, EV charging stations, storage facilities, etc), which are represented by an aggregator and a DSO, noted as participant 0 of set N. Each facility $n \in N/\{0\}$ is located at a

particular node of the DSO's distribution network and can perform DR actions. Continuous time is divided into a set T of timeslots for a given horizon ahead.

Each facility is able to control the power consumption of its flexibility assets through the facility's EMS. The set of flexibility assets of facility $\boldsymbol{n}$ is denoted by $\boldsymbol{\Gamma_n}$. The aggregated energy consumption of a facility at timeslot $\boldsymbol{t}$ is denoted by $\boldsymbol{x_{n,t}}$, while the energy consumption of a particular asset $\boldsymbol{\gamma} \in \boldsymbol{\Gamma_n}$ of facility $\boldsymbol{n}$, is denoted by $\boldsymbol{p_{\gamma,t}}$. Therefore, we have

$$\sum_{\boldsymbol{\gamma}\in\boldsymbol{\Gamma_n}} \boldsymbol{p_{\gamma,t}} = \boldsymbol{x_{n,t}}, \quad \forall \boldsymbol{n} \in \boldsymbol{N}/\{\boldsymbol{0}\}, \boldsymbol{t} \in \boldsymbol{T}. \qquad (4.1)$$

Each facility features a set $\mathcal{Y}_n$ of local variables, which includes $x_{n,t}, p_{\gamma,t}$ (for every t and γ) and also other local variables, depending on the particular models of the facility's flexibility assets. A facility also bears a set C_n of feasible operational points, defined by a number of operational constraints on the combinations $\{y\}_{y\in\mathcal{Y}_n}$ of all local variables $y \in \mathcal{Y}_n$. Therefore, we have:

$$\{\boldsymbol{y}\}_{\boldsymbol{y}\in\boldsymbol{\mathcal{Y}_n}} \in \boldsymbol{C_n}, \quad \forall \boldsymbol{n} \in \boldsymbol{N}/\{\boldsymbol{0}\}. \qquad (4.2)$$

Detailed asset models are presented in section 4.6 below. While those models facilitate the adequate evaluation of the proposed DROaaS architecture, the architecture is open and transparent to the facility DR models used, in the sense that it is not bounded to those, or any other, particular models. For this reason, the operational constraints (cf. 4.2) are kept in an abstract and general form for now.

The DSO is responsible for maintaining the distribution network within safe operational limits. Assuming a radial network, let $\boldsymbol{A}$ denote the set of network nodes and $\boldsymbol{B}$ the set of branches. For a node $\boldsymbol{a} \in \boldsymbol{A}$, let $\boldsymbol{\Omega_p(a)}$ (or $\boldsymbol{\Omega_d(a)}$) denote the set of predecessor (or descendant, respectively) nodes connected to node $\boldsymbol{a}$. In node $\boldsymbol{a}$, there is a certain amount of power consumption $\mathbf{P}^{\mathbf{d}}_{\boldsymbol{a,t}}$, as well as a RES power generation $\mathbf{P}^{\mathbf{RES}}_{\boldsymbol{a,t}}$, which the DSO can choose to curtail by a factor of $\boldsymbol{1-\theta_{a,t}} \in [\boldsymbol{0},\boldsymbol{1}]$, where $\boldsymbol{\theta_{a,t}} = \boldsymbol{1}$ means that there is no RES curtailment and $\boldsymbol{\theta_{a,t}} = \boldsymbol{0}$ means that the whole RES output of node $\boldsymbol{a}$ is curtailed. Finally, let $\boldsymbol{N_a}$ denote the set of facilities located at node $\boldsymbol{a} \in \boldsymbol{A}$. Towards modeling the flows and constraints of the physical electricity grid, we use the linearized DistFlow equations [59], defined by the following set of constraints:

$$\sum_{\boldsymbol{i}\in\boldsymbol{\Omega_p(a)}} \boldsymbol{P_{ia,t}} - \sum_{\boldsymbol{n}\in\boldsymbol{N_a}} \boldsymbol{x_{n,t}} - \mathbf{P}^{\mathbf{d}}_{\boldsymbol{a,t}} + \boldsymbol{\theta_{a,t}}\mathbf{P}^{\mathbf{RES}}_{\boldsymbol{a,t}} = \sum_{\boldsymbol{j}\in\boldsymbol{\Omega_d(a)}} \boldsymbol{P_{aj,t}}, \quad \forall \boldsymbol{a} \in \boldsymbol{A}, \boldsymbol{t} \in \boldsymbol{T} \qquad (4.3)$$

$$\sum_{i\in\Omega_p(a)} Q_{ia,t} - \sum_{n\in N_a} \mathrm{f}_{n,t}\, x_{n,t} - \mathrm{Q}^{\mathrm{d}}_{a,t} + \theta_{a,t}\mathrm{f}^{\mathrm{RES}}_{a,t}\mathrm{P}^{\mathrm{RES}}_{a,t} = \sum_{j\in\Omega_d(a)} Q_{aj,t}, \quad \forall a \in A, t \in T \qquad (4.4)$$

$$V_{a,t} = V_{i,t} - 2\left(\mathrm{R}_{ia}P_{ia,t} + \mathrm{X}_{ia}Q_{ia,t}\right), \quad \forall a \in A, i \in \Omega_p(a), t \in T \qquad (4.5)$$

$$\underline{V}_a \le V_{a,t} \le \overline{V}_a, \qquad \forall a \in A, t \in T$$
$$\underline{P}_{aj,t} \le P_{aj,t} \le \overline{P}_{aj,t}, \qquad \forall aj \in B, t \in T \qquad (4.6 – 4.8)$$
$$\underline{Q}_{aj,t} \le Q_{aj,t} \le \overline{Q}_{aj,t}, \qquad \forall aj \in B, t \in T,$$

where $f_{n,t}$ are the parameters relating active and reactive power (through the power factor), and $P_{ia,t}$ and $Q_{ia,t}$ are the active and reactive power flowing on the branch ia connecting nodes $i \in A$ and $a \in A$ of the distribution network.

Equations (4.3) and (4.4) represent the active and reactive power balances at each distribution node. Namely, the total outgoing power $\sum_{j \in \Omega_d(a)} \boldsymbol{P_{aj,t}}$ from node $\boldsymbol{a}$, equals the incoming power $\sum_{i \in \Omega_p(a)} \boldsymbol{P_{ia,t}}$ minus the net power consumption $\sum_{n \in N_a} \boldsymbol{x_{n,t}} + \mathbf{P}^{\mathbf{d}}_{a,t} - \boldsymbol{\theta_{a,t}} \mathbf{P}^{\mathbf{RES}}_{a,t}$ of node $\boldsymbol{a}$. Equation (4.5) describes the voltage drop between each pair of neighboring nodes $\boldsymbol{a}, \boldsymbol{i}$ where $\boldsymbol{i} \in \boldsymbol{\Omega_p(a)}$. Variable $\boldsymbol{V_{a,t}}$ denotes the squared voltage of node $\boldsymbol{a}$ at $\boldsymbol{t}$, while $\mathbf{R_{ia}}$ and $\mathbf{X_{ia}}$ are the resistance and reactance, respectively, of branch $\boldsymbol{ia}$. The grid's voltages and active/reactive power flows must satisfy certain limits to ensure the physical grid's operational safety. Constraints (4.6) make sure that voltages in all nodes stay within safe margins, while (4.7) and (4.8) limit the active and reactive power flows for all branches.

4.4.2 DR problem formulation

The DSO decides the amount of RES curtailments $\theta_{a,t}$, that come at a cost of c^{curt}_a per 1 MW of RES generation curtailment, as well as variables $x_{0,t}$ that express the power exchange with the main grid. A cost function $d_0(\mathcal{Y}_0)$ is defined for the DSO, to capture the cost of exchanging energy with the main grid and the cost of RES curtailments, namely:

$$d_0(\mathcal{Y}_0) = \sum_{t \in T} \left(\pi_t x_{0,t} + \sum_{a \in A} (1 - \theta_{a,t}) \mathrm{P}^{\mathrm{RES}}_{a,t} \mathrm{c}^{\mathrm{curt}}_a \right) \qquad (4.9)$$

where π_t the price for importing/exporting energy and $\mathcal{Y}_0 = \{P_{ia}, P_{aj}, Q_{ia}, Q_{aj}, V_{a,t}, x_{0,t}, \theta_{a,t}\}$

On the other hand, each facility bears a DR-cost function $\boldsymbol{d_n}(\boldsymbol{\mathcal{Y}_n})$, where $\boldsymbol{\mathcal{Y}_n}$ denotes the set of local variables of the facility. The function $\boldsymbol{d_n}(\cdot)$ is used by the facility to model the DR costs of its assets. For example, an EV charging station would need to compensate its EV users (or offer price discounts) to counteract their dis-satisfaction for suffering delays in their battery charging due to congestion in the electricity network. Similarly to constraints (4.2), facility DR cost functions are kept in a general form in this section, but they are modeled explicitly in section 4.6 below for the performance evaluation tests.

The objective of the market clearing algorithm is to make sure that the network operates within the feasible operational area, while minimizing the aggregate system cost (i.e., the cost of DR actions and the cost of exchanging power with the main grid), as follows:

$$\mathbf{min}\{\textstyle\sum_{n \in \mathbf{N}} \boldsymbol{d_n}(\boldsymbol{y_n})\}, \text{ s.t. } (4.1) – (4.9) \qquad (4.10)$$

The intuition behind problem (4.10) is that, in case the satisfaction of the physical grid's safety constraints necessitates DR actions, RES curtailments and/or power imports, the DSO will decide the least expensive combination of actions. For example, facilities that have a very small DR cost (e.g., a battery or a very flexible load) will be prioritized for dispatch actions before modifying the consumption profile of critical loads or resorting to RES curtailments and/or power imports at times where the electricity prices are high.

4.4.3 DR problem decomposition

Solving problem (4.10) directly, in a centralized fashion, poses a number of challenges. The first challenge is that all models (DR cost functions and operational constraints) of the facilities would need to be communicated to a central entity, which raises security and privacy concerns. A second issue is that the large number of variables makes the problem computationally intensive.

In order to overcome these issues, problem (4.10) can be solved in a distributed fashion using a Lagrangian decomposition. Each facility solves a local optimization problem to decide the value of its local variables $\boldsymbol{y_n}$, while the DSO solves an optimal power flow problem. The procedure iterates, while coordination is achieved by updating a set of Lagrange multipliers $\boldsymbol{\lambda_{a,t}}$ and $\boldsymbol{\mu_{a,t}}$ that are related to the dual variables of the active and reactive power balance constraints, respectively. More specifically, we consider the Alternating Direction Method of Multipliers (ADMM). By taking the augmented Lagrangian of problem (4.10), we have:

$$\begin{aligned}\mathcal{L} = \sum_{n\in N} \boldsymbol{d_n}(\boldsymbol{y_n}) + \sum_{a\in A}\sum_{t\in T}\left(\boldsymbol{\lambda_{a,t}}\boldsymbol{g_{a,t}} + \frac{\rho_1}{2}\| \boldsymbol{g_{a,t}}\|^2\right) \\ + \sum_{a\in A}\sum_{t\in T}\left(\boldsymbol{\mu_{a,t}}\boldsymbol{h_{a,t}} + \frac{\rho_2}{2}\| \boldsymbol{h_{a,t}}\|^2\right)\end{aligned} \tag{4.11}$$

where

$$\begin{aligned}\boldsymbol{g_{a,t}} = \sum_{i\in\Omega_p(a)} \boldsymbol{P_{ia,t}} + \sum_{n\in N_a} \boldsymbol{x_{n,t}} + \mathbf{P}^{\mathbf{d}}_{\boldsymbol{a,t}} - \boldsymbol{\theta_{a,t}}\mathbf{P}^{\mathbf{RES}}_{\boldsymbol{a,t}} \\ - \sum_{j\in\Omega_d(a)} \boldsymbol{P_{aj,t}}\end{aligned} \tag{4.12}$$

$$\begin{aligned}h_{a,t} = \sum_{i\in\Omega_p(a)} Q_{ia,t} + \sum_{n\in N_a} \mathrm{f}_{n,t}\, x_{n,t} + \mathrm{Q}^{\mathrm{d}}_{a,t} \\ -\theta_{a,t}\mathrm{f}_{a,t}\mathrm{P}^{\mathrm{RES}}_{a,t} - \sum_{j\in\Omega_d(a)} Q_{aj,t}\,.\end{aligned} \tag{4.13}$$

An iterative method for solving problem (4.10) is defined based on the following variable update rules:

Facility

$$\{y\}^{(k+1)}_{y\in\mathcal{Y}_n} = \underset{\mathcal{Y}_n}{\operatorname{argmin}}\left\{\mathcal{L}^{(k)}\right\} \tag{4.14}$$

s.t. (4.1 – 4.2)

DSO

$$P_{ia}^{(k+1)}, P_{aj}^{(k+1)}, Q_{ia}^{(k+1)}, Q_{aj}^{(k+1)}, V_{a,t}^{(k+1)}, \theta_{a,t}^{(k+1)} = \text{argmin}_{P_{ia}, P_{aj}, Q_{ia}, Q_{aj}, V_{a,t}, \theta_{a,t}} \left\{ \mathcal{L}^{(k+1)} \right\} \quad \text{(4.15)}$$

s.t. (4.5) – (4.9)

Coordinating entity (i.e. aggregator)

$$\boldsymbol{\lambda}_{a,t}^{(k+1)} = \boldsymbol{\lambda}_{a,t}^{(k)} + \boldsymbol{\rho}_1 \boldsymbol{g}_{a,t}^{(k+1)}$$
$$\boldsymbol{\mu}_{a,t}^{(k+1)} = \boldsymbol{\mu}_{a,t}^{(k)} + \boldsymbol{\rho}_2 \boldsymbol{h}_{a,t}^{(k+1)} \quad \text{(4.16 – 4.17)}$$

where ρ_1 and ρ_2 are step update coefficients.

This formulation allows problem (4.10) to be parallelized in order to be solved by appropriate computational resources in a coordinated distributed fashion. In particular, the computing task of each facility, i.e. solving problem (4.14), can be viewed as a self-dispatch problem where the facility decides the power consumption of each asset under the current active and reactive electricity prices $\boldsymbol{\lambda}_{a,t}^{(k)}, \boldsymbol{\mu}_{a,t}^{(k)}$ of the facility's node. On the other hand, the DSO solves an optimal power flow problem, i.e. problem (4.15), to decide whether any RES generation needs to be curtailed as well as the amount of power exchange with the main grid. The sequence and variable exchange among the execution nodes that execute each function is described in Algorithm 1 that is shown below.

Algorithm 1: Iterative distributed algorithm for solving problem (10)

1 Initialize $k = 0, \lambda_{a,t}^{(0)} = 0, \mu_{a,t}^{(0)} = 0$;
2 **while** $g_{a,t}, h_{a,t} \geq \varepsilon$ **do**
3 Multipliers $\lambda_{a,t}^{(k)}, \mu_{a,t}^{(k)}$ are communicated to all computational resources;
4 The resource responsible for facility n solves problem (14);
5 The solutions are communicated to the computational resource responsible for the resource responsible for the DSO, which solves (15);
6 The solutions are communicated to the computational resource responsible for the multiplier and iteration updates;
7 Multipliers are updated based on (16), (17) and $k = k + 1$.

Figure 22: Iterative distributed algorithm for solving problem (4.10)

4.5 Algorithmic solution for cloud resource allocation

We consider the functionality of the proposed DROaaS framework, as a means to efficiently coordinate the calculations of a set R of DR requests (corresponding to different distribution networks), where a DR request $r \in R$ is an instance of problem (4.10) that is solved through Algorithm 1. Each particular DR request is decomposed into $N+1$ subproblems, as presented in the previous section. Each subproblem corresponds to the local optimization problem of a facility (or the DSO) and can be viewed as a different computation task. Therefore, each DR request $r \in R$ is characterized by its arrival time α_r, its set of tasks F_r (facilities and DSO) and an upper bound $\overline{\text{lat}}_r$ on the allowable latency per iteration.

Each task requires the transmission of input data $\boldsymbol{\delta_f^{in}}$ and output data $\boldsymbol{\delta_f^{out}}$. These tasks can be executed in the facilities (i.e. edge resources) that they originate from, or be forwarded to aggregation points (i.e. fog resources) or to the cloud, and they introduce a latency $\boldsymbol{lat_f}$. The incentive for moving tasks from the edge (on site) to the fog (other sites such as the aggregator's premises) and to the cloud (central sites) is the lower capacity of lower-level resources that may prolong the execution time of the set of tasks beyond the allowable delay. Aggregating multiple tasks in fog resources (e.g. aggregator's computing infrastructure) can reduce the overall cost of the operation, assuming that resources' marginal cost reduces as a function of the submitted workload. On the other hand, moving tasks to higher layers of the COMNET infrastructure introduces networking latency that may increase the tasks' overall execution time, the so-called makespan.

Let graph $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ jointly represent the flexibility markets and the computing and networking (COMNET) infrastructure. The set $\boldsymbol{V} = \boldsymbol{V_c} \cup \boldsymbol{V_f}$ consists of the nodes $\boldsymbol{V_c}$ that possess computational resources, and the nodes $\boldsymbol{V_f}$: $\boldsymbol{f} \in \boldsymbol{F_r}$, $\boldsymbol{r} \in \boldsymbol{R}$ where the facilities are connected. Facility nodes can be also equipped with processing units, thus $\boldsymbol{V_c} \cap \boldsymbol{V_f} \neq \varnothing$ in general. The set $\boldsymbol{E}$ corresponds to virtual links that interconnect the nodes over wired and wireless communication paths. Let $\boldsymbol{M}$ denote the set of types of computational resources available in the system. A resource of type $\boldsymbol{m} \in \boldsymbol{M}$, located at node $\boldsymbol{v} \in \boldsymbol{V}$, is characterized by a processing cost $\mathbf{c}_{\boldsymbol{v},\boldsymbol{m}}^{\mathbf{proc}}$. Each virtual link $\boldsymbol{i}, \boldsymbol{j} \in \boldsymbol{E}$ is characterized by a network cost $\mathbf{c}_{\boldsymbol{i},\boldsymbol{j}}^{\mathbf{net}}$, depending on its available networking capacity. The overall transmission rate of the virtual link $\boldsymbol{i}, \boldsymbol{j}$ is denoted as $\mathbf{tr}_{\boldsymbol{i},\boldsymbol{j}}$, resulting in a transmission latency for the data that needs to be transferred between nodes $\boldsymbol{i}$ and $\boldsymbol{j}$ and a propagation latency that depends on the physical distance of the virtual link.

The goal of the resource optimization procedure is to minimize the weighted sum of the processing per iteration cost and the latency for serving all the DR requests in $\boldsymbol{R}$, while respecting the time constraints of each request. In the next subsections we formulate the problem as an Integer Linear Program (ILP), and also provide a heuristic algorithm for keeping the computational time low.

4.5.1 Optimal ILP for the allocation of computational resources

In what follows, we present the ILP formulation of the dynamic resource allocation problem. We use the index f to refer to a facility (and respective computation task) of any DR request, i.e. $f \in F$, where $F = \bigcup_{r \in R} \{F_r\}$. Let binary variable $\zeta_{v,m,f}$ denote whether a

virtual machine of type m located at node v, is assigned to perform the calculation task f. A resource m at node v needs $\mathrm{pr}_{v,m,f}$ time to perform the computations of task f.

To speed-up the calculations, we make use of a pre-processing phase, in which we pre-calculate $\boldsymbol{\kappa}$ shortest paths with length $\mathbf{l}_{i,j,\kappa}$ between each pair of nodes $\boldsymbol{i},\boldsymbol{j} \in \boldsymbol{E}$, which include the paths from the location $\boldsymbol{v_f}$ of each facility $\boldsymbol{f}$, to the different processing nodes $\boldsymbol{v} \in \boldsymbol{V}$ and between the processing nodes. Given the communication network topology $\boldsymbol{G}$, let $\boldsymbol{K_{i,j}}$ denote the set of $\boldsymbol{\kappa}$ shortest paths between nodes $\boldsymbol{i},\boldsymbol{j}$, and set $\boldsymbol{\Lambda_{i,j}}$ contain their respective lengths $\mathbf{l}_{i,j,\kappa}$. Then, binary variable $\boldsymbol{\beta_{v_f,v,\kappa}}$ denotes whether the corresponding facility of task $\boldsymbol{f}$ (located at node $\boldsymbol{v_f}$) is connected to a node $\boldsymbol{v}$ over virtual link $\boldsymbol{\kappa} \in \boldsymbol{K_{v_f,v}}$ or not.

An integer variable, denoted as $\boldsymbol{\psi_{v,m,f}}$, indicates the timeslot[10] in which a resource of type $\boldsymbol{m}$, located at node $\boldsymbol{v}$, starts the processing of task $\boldsymbol{f}$. Finally, binary variable $\boldsymbol{\xi_{v,m,f,\hat{f}}}$ denotes whether the calculation of task $\boldsymbol{f}$ at $\boldsymbol{m},\boldsymbol{v}$ is performed before that of task $\boldsymbol{\hat{f}}$. The objective of optimal resource allocation is to minimize the overall processing and network costs, i.e.:

$$\min_{\mathcal{W}} \left\{ \mathrm{w} \cdot \sum_{v\in V}\sum_{m\in M}\sum_{f\in F} \zeta_{v,m,f} \cdot \mathrm{c}_{v,m}^{\mathrm{proc}} + (1-\mathrm{w}) \cdot \sum_{v\in V}\sum_{m\in M}\sum_{f\in F} (\psi_{v,m,f} + \zeta_{v,m,f} \cdot \mathrm{pr}_{v,m,f}) \right\} \quad (4.18)$$

where $\mathcal{W} = \{\zeta_{v,m,f}, \beta_{v_f,v,\kappa}, \psi_{v,m,f}, \xi_{v,m,f,\hat{f}}\}$ and w is an objective weighting coefficient taking values between 0 and 1. When $\mathrm{w} = 0$, the latency for serving the DR requests is minimized, while when $\mathrm{w} = 1$, the processing per iteration cost is minimized. In intermediary cases, where cost and latency are traded off, the value of w needs to be appropriately tuned, so that the processing cost (measured in monetary units) is balanced with the latency value (measured in units of time). The optimization is subject to the following constraints. Each task has to be assigned to exactly one virtual machine:

$$\sum_{\boldsymbol{v}\in\boldsymbol{V}} \sum_{\boldsymbol{m}\in\boldsymbol{M}} \boldsymbol{\zeta_{v,m,f}} = \mathbf{1},\ \forall \boldsymbol{f} \in \boldsymbol{F}. \quad (4.19)$$

In order to assign task f to resource v,m, a connection path must be selected:

$$\sum_{\boldsymbol{\kappa}\in\boldsymbol{K_{v_f,v}}} \boldsymbol{\beta_{v_f,v,\kappa}} \geq \sum_{\boldsymbol{m}\in\boldsymbol{M}} \boldsymbol{\zeta_{v,m,f}},\ \forall \boldsymbol{f} \in \boldsymbol{F}, \boldsymbol{v} \in \boldsymbol{V}. \quad (4.20)$$

We assume that the multiplier updates are made by the aggregator itself. Let $\tilde{v}_r$ denote the node where the DSO of DR request r is located and $\tilde{f}_r$ denote the special task of multiplier update. Each facility allocates a virtual link for forwarding the data to the DSO

[10] The set T of timeslots defined in section 4.4.1 refers to operational timeslots e.g. of 15-minute duration. On the contrary, here we refer to timeslots that relate to the execution times of the calculations. Those are of much smaller durations. In fact, these timeslots belong to a set $\mathcal{T}$, where the total duration of all timeslots in $\mathcal{T}$, is smaller than the duration of one timeslot $t \in T$, in order to satisfy the requirement that the calculations' execution should finish before the operational timeslot changes.

$$\sum_{\kappa \in K_{v_f,\tilde{v}_r}} \boldsymbol{\beta}_{v_f,\tilde{v}_r,\kappa} = \mathbf{1},\ \forall \boldsymbol{f} \in \boldsymbol{F_r}, \boldsymbol{r} \in \boldsymbol{R}. \quad (4.21)$$

Node v cannot begin the execution of task f before receiving the input data δ_f^{in} of f. This is subject to transmission and propagation delays, and the starting time of task f's at m, v can be calculated to be:

$$\psi_{v,m,f} \geq \frac{\beta_{v_f,v,\kappa} \mathrm{l}_{v_f,v,\kappa}}{\Phi} + \frac{\delta_f^{\mathrm{in}}}{\mathrm{tr}_{v_f,v}} - (1 - \sum_{m \in M} \zeta_{v,m,f}) \cdot \mathrm{Q}$$
$$\forall m \in M, v \in V, f \in F, \quad (2$$ (4.22)

where Φ is the speed of light and Q is a sufficiently big number. The aggregator cannot update the multipliers before receiving the response of each calculation task, implying that:

$$\psi_{\tilde{v}_r,m,\tilde{f}_r} \geq \psi_{v,m,f} + \mathrm{pr}_{v,m,f} + \frac{\beta_{\tilde{v}_r,v_f,\kappa} \mathrm{l}_{\tilde{v}_r,v_f,\kappa}}{\Phi} + \frac{\delta_f^{\mathrm{out}}}{\mathrm{tr}_{\tilde{v}_r,v_f}}$$
$$- (1 - \sum_{m \in M} \zeta_{\tilde{v}_r,m,\tilde{f}_r}) \cdot \mathrm{Q},$$
$$\forall m \in M, v \in V, f \in F_r, r \in R. \quad (2$$ (4.23)

When the multipliers are updated, an iteration of the distributed algorithm is completed and the respective latency per iteration constraint must be satisfied:

$$\boldsymbol{\psi}_{\tilde{v}_r,m,\tilde{f}_r} + \mathbf{pr}_{\tilde{v}_r,m,\tilde{f}_r} \leq \overline{\mathbf{lat}}_r, \quad \forall \boldsymbol{m} \in \boldsymbol{M}, \boldsymbol{v} \in \boldsymbol{V}, \boldsymbol{f} \in \boldsymbol{F_r}, \boldsymbol{r} \in \boldsymbol{R}. \quad (4.24)$$

Finally, the following three constraints ensure that the execution time ordering is preserved and there are no overlaps (i.e., simultaneous task executions at the same machine):

$$\boldsymbol{\xi}_{v,m,f,\hat{f}} + \boldsymbol{\xi}_{v,m,\hat{f},f} = \mathbf{1}, \quad \forall \boldsymbol{m} \in \boldsymbol{M}, \boldsymbol{v} \in \boldsymbol{V}, \boldsymbol{f}, \hat{\boldsymbol{f}} \in \boldsymbol{F}, \boldsymbol{f} \neq \hat{\boldsymbol{f}} \quad (4.25)$$

$$\psi_{m,v,f} + \mathrm{pr}_{v,m,f} - \psi_{m,v,\hat{f}} \leq \left(1 - \xi_{v,m,f,\hat{f}} + 2 - \zeta_{v,m,f} - \zeta_{v,m,\hat{f}}\right) \cdot \mathrm{Q}$$
$$\forall m \in M, v \in V, f, \hat{f} \in F, f \neq \hat{f} \quad (4.26)$$

$$\psi_{m,v,\hat{f}} + \mathrm{pr}_{m,v,\hat{f}} - \psi_{m,v,f} \leq \left(1 - \xi_{v,m,\hat{f},f} + 2 - \zeta_{v,m,\hat{f}} - \zeta_{v,m,f}\right) \cdot \mathrm{Q}$$
$$\forall m \in M, v \in V, f, \hat{f} \in F, f \neq \hat{f}. \quad (4.27)$$

For large instances, the optimal ILP solution can take a long time to calculate. Thus, in the next subsection, we present a heuristic algorithm that can achieve a near-optimal solution with minimal computational time.

4.5.2 Heuristic algorithm for fast resource allocation

The heuristic algorithm decomposes the selection of processing nodes and transmission links in a separable form, assuming there is no coupling between the nodes where the processing is performed and the node which is responsible for communicating the multipliers and the iteration update. Thus, the problem can be efficiently solved by solving

the two decoupled problems, where the first finds the pairing of facility tasks to processing nodes, and the second finds the pairings of facility nodes to the node responsible for the iterations. Since the number of variables and constraints in this case is not large, many algorithms can be applied. The heuristic, presented in Algorithm 2 below, is based on relaxing the grouping constraints by first assigning facility tasks to processing nodes and then DR requests to the communicating node.

More specifically, the heuristic algorithm serves the DR requests sequentially, one by one. To do so, the DR requests are sorted in descending order based on their service latency requirements (line 1). Hence, decisions for the DR requests with strict latency requirements are prioritized. Then, the facility tasks of the request are examined sequentially, and resources are allocated based on a best fit approach and the selected objective function (lines 3-4). The allocation of each facility task is determined (and updated) by solving problem (4.18) – (4.27), but keeping all the variables of other tasks fixed. Since the DSO's task has not been allocated at this point (i.e., the DSO variables are initialized to zero), constraint (4.23) is relaxed to prevent infeasibility. After all facility tasks are allocated, the algorithm allocates resources to the DSO task (line 5), by solving (4.18) – (4.27), while keeping the variables of facility tasks fixed.

Algorithm 2: DROaaS heuristic algorithm

Input: $G=(V,E)$, M, R, w, $\{c^{\text{proc}}_{v,m}\}_{\forall v\in V, m\in M}$, $\{c^{\text{net}}_{i,j}\}_{\forall (i,j)\in E}$

Output: Allocation of facility tasks to computational resources: $\{\zeta_{v,m,f}, \psi_{v,m,f}\}_{\forall v,m,f}$

Initialize: $\{\zeta_{v,m,f}, \psi_{v,m,f}\}_{\forall v,m,f} = 0$

1 Sort the DR requests in descending order of their latency bound $\overline{\text{lat}}_r$;

2 **for** *each DR request* $r \in R$ **do**

3 **for** *each facility task* $f \in F_r$ **do**

4 Update $\zeta_{v,m,f}, \psi_{v,m,f}$, by solving problem (18)-(27) with $\{\zeta_{v,m,\hat{f}}, \psi_{v,m,\hat{f}}\}_{\forall v,m,\hat{f}:\hat{f}\neq f}$ fixed, and with constraint (23) relaxed

  end

5 Update $\zeta_{v,m,\tilde{f}}, \psi_{v,m,\tilde{f}}$, by solving problem (18)-(27) to allocate the DSO tasks, with the allocation $\{\zeta_{v,m,f}, \psi_{v,m,f}\}_{\forall v,m,f:f\neq\tilde{f}}$ of the facilities' tasks fixed

end

Figure 23: The proposed DROaaS heuristic algorithm

4.6 Simulation setup

In this section we present the simulation setup, which includes a set of detailed heterogeneous facility DR models, a benchmark DSO network, and COMNET infrastructure.

4.6.1 Facility DR models

Table 5: Technical characteristics for all types of facilities

Parameters	Values
Curtailable loads	
$\underline{x}_\gamma$ (kW)	0
$\overline{x}_\gamma$ (kW)	[1, 5]
$\tilde{x}_\gamma$ (%$\overline{x}_\gamma$)	[70, 100]
c_γ (€/kW^2)	[0.25, 0.50]
Curtailable loads With Ramp Constraints	
$\underline{x}_\gamma$ (kW)	0
$\overline{x}_\gamma$ (kW)	[10, 15]
c_γ (€/kW^2)	[0.03, 0.05]
r_γ^{down} (kW)	[4, 6]
r_γ^{up} (kW)	[4, 6]
Time-shiftable Loads	
$\underline{x}_\gamma$ (kW)	0
$\overline{x}_\gamma$ (kW)	[2, 12]
E_γ (kWh)	[6, 36]
t_γ^{arr} (h)	[3, 9]∪[14, 20]
t_γ^{dep} (h)	[5, 11]∪[16, 22]
t_γ (h)	$t_\gamma^{arr} + E_\gamma/\overline{x}_\gamma$
s_γ (€· h)	[1.0, 1.1]
Fully Flexible Loads	
$\underline{x}_\gamma$ (kW)	0
$\overline{x}_\gamma$ (kW)	2
t_γ^{arr} (h)	[1, 21]
t_γ^{dep} (h)	[1, 21] + [2, 5]
$\underline{E}_\gamma$ (kWh)	[4.8, 5.1]
$\overline{E}_\gamma$ (kWh)	[5.5, 6.0]
l_γ^1 (€/kW)	[0.25, 0.75]
l_γ^2 (€)	[2, 3]
Storage Facilities	
$\underline{x}_\gamma$ (kW)	0
$\overline{x}_\gamma$ (kW)	2.5
e_γ^c (%)	95
e_γ^d (%)	95
$\overline{SOC}_\gamma$ (kWh)	5
$SOC_{\gamma,0}$ (kWh)	2.5
b_γ (#)	2
Thermostatically Controlled Loads	
$\underline{x}_\gamma$ (kW)	0
$\overline{x}_\gamma$ (kW)	5
$\underline{H}_\gamma$ (° F)	74
$\overline{H}_\gamma$ (° F)	83
$\tilde{H}_\gamma$ (° F)	79
h_γ^{ins} (%)	90
h_γ^{eff} (° F/kW)	-3
h_γ^{cost} (€/ ° F^2)	[0.15, 0.20]

We consider several heterogeneous facility DR models, where the differences lie in the modeling choices of the facility manager entity or in the nature of the facility's flexible loads. All flexible loads are characterized by minimum and maximum operational points between which the load's electricity consumption must lie, i.e.,

$$\underline{\mathbf{x}}_{\boldsymbol{\gamma}} \leq \boldsymbol{x}_{\boldsymbol{\gamma},t} \leq \overline{\mathbf{x}}_{\boldsymbol{\gamma}}, \quad \forall \boldsymbol{\gamma} \in \boldsymbol{\Gamma}_{\boldsymbol{n}}, \boldsymbol{n} \in \boldsymbol{N} \qquad (4.28)$$

The DR cost d_n of facility n is defined as the sum of the DR costs $d_{\gamma,n}$ of all the assets that it operates, i.e.

$$\boldsymbol{d_n(\mathcal{Y}_n) = \sum_{\gamma \in \Gamma_n} d_{\gamma,n}(\mathcal{Y}_n), \quad \forall n \in N.} \qquad (4.29)$$

In the following subsections, we present the facility DR models used in the simulations. A facility DR model refers to the specific formulation of the facility's constraints (4.2) and its DR cost function $d_n(\mathcal{Y}_n)$. Based on its DR model, each facility type bears different computational requirements for solving its local problem (4.14), which greatly interferes with the resource allocation problem. The particular values for all parameters used in the following subsections are presented in the table above.

4.6.1.1 Facility with curtailable loads

The set of facilities belonging to this type is denoted by N_{curt}. A curtailable load $\gamma \in \Gamma_{n:n\in N_{\text{curt}}}$ has a desired consumption $\tilde{\text{x}}_{\gamma,t}$ at timeslot t and is characterized by a set of DR cost parameters $\text{c}_{\gamma,t}$ that relate to the level of the load's inelasticity. For these loads, the set of controllable variables consists only of the electricity consumption variables $x_{\gamma,t}$, i.e. $\mathcal{Y}_{n:n\in N_{\text{curt}}} = \{x_{\gamma,t}\}_{\gamma\in\Gamma_n, t\in T}$. The DR cost function of a load, as adapted by [64] and [65], is defined by:

$$\boldsymbol{d_{\gamma,n}(\mathcal{Y}_n) = \sum_{t\in T} d_{\gamma,n,t}, \quad \forall \gamma \in \Gamma_n, n \in N_{\text{curt}},} \qquad (4.30)$$

where

$$\boldsymbol{d_{\gamma,n,t} = \text{c}_{\gamma,t}\left(\tilde{\text{x}}_{\gamma,t} - x_{\gamma,t}\right)^2, \quad \forall \gamma \in \Gamma_n, n \in N_{\text{curt}}.} \qquad (4.31)$$

4.6.1.2 Facility with curtailable loads and ramp constraints

For some assets it might be relevant to constraint the ramp up/down rates $\text{r}^{\text{up}}/\text{r}^{\text{down}}$ of energy consumption, in order to avoid abrupt changes in their consumption from one timeslot to the next. For this type of loads $\gamma \in \Gamma_n, n \in N_{\text{ramp}}$, the control variables, constraints and cost function are the same as those of curtailable loads, but with an additional time-coupling constraint:

$$\boldsymbol{\text{r}_\gamma^{\text{down}} \le x_{\gamma,t} - x_{\gamma,t-1} \le \text{r}_\gamma^{\text{up}}, \quad \forall \gamma \in \Gamma_n, n \in N_{\text{ramp}}.} \quad (4.32)$$

4.6.1.3 Facility with time-shiftable loads

This set of facilities is denoted by N_{shift}. A load $\gamma \in \Gamma_{n:n\in N_{\text{shift}}}$ has a desired energy consumption E_γ that must be fulfilled within the time interval $\left[\text{t}_\gamma^{\text{arr}}, \text{t}_\gamma^{\text{dep}}\right]$:

$$\boldsymbol{\sum_{t\in\left[\text{t}_\gamma^{\text{arr}}, \text{t}_\gamma^{\text{dep}}\right]} x_{\gamma,t} = \text{E}_\gamma, \quad \forall \gamma \in \Gamma_n, n \in N_{\text{shift}}} \qquad (4.33)$$

The only decision variables are again $x_{\gamma,t}$. The load also has a desired completion time $\tilde{t}_\gamma \leq t_\gamma^{dep}$. If part of the load's required energy consumption is consumed after $\tilde{t}_\gamma$, then the load bears a cost, defined as:

$$\boldsymbol{d_{\gamma,n}(\mathcal{Y}_n) = \sum_{t\in T} d_{\gamma,n,t}}, \quad \forall \boldsymbol{\gamma \in \Gamma_n, n \in N_{shift}}, \tag{4.34}$$

where

$$\boldsymbol{d_{\gamma,n,t} = \sum_{t=\tilde{t}_\gamma+1}^{t_\gamma^{dep}} \frac{s_\gamma^{(t-\tilde{t}_\gamma)}}{E_\gamma} x_{\gamma,t}, \ \gamma \in \Gamma_n, \forall n \in N_{shift}} \tag{4.35}$$

Intuitively, the term $s_\gamma^{(t-\tilde{t}_\gamma)}$ imposes a higher DR cost for later timeslots through the exponent, while parameter $s_\gamma \geq 1$ relates to the load's inelasticity. This model is adapted from [66].

4.6.1.4 Facility with fully flexible loads

This set of facilities is denoted by N_{flex}. A load $\gamma \in \Gamma_{n:n\in N_{\text{flex}}}$ is characterized by a feasible time interval $\left[t_\gamma^{arr}, t_\gamma^{dep}\right]$, as well as a desired energy consumption $\tilde{E}_\gamma$ and a minimum acceptable energy consumption $\underline{E}_\gamma$ for that time interval, i.e.

$$\boldsymbol{\underline{E}_\gamma \leq \sum_{t\in\left[t_\gamma^{arr}, t_\gamma^{dep}\right]} x_{\gamma,t} \leq \tilde{E}_\gamma}, \quad \forall \boldsymbol{\gamma \in \Gamma_n, n \in N_{flex}}. \tag{4.36}$$

The set of controllable variables is again $\mathcal{Y}_{n:n\in N_{\text{flex}}} = \{x_{\gamma,t}\}_{\gamma\in\Gamma_n, t\in T}$. The DR cost of a flexible load of this type, adapted from [67], is defined as:

$$\boldsymbol{d_{\gamma,n}(\mathcal{Y}_n) = l_\gamma^1 \sum_{t\in\left[t_\gamma^{arr}, t_\gamma^{dep}\right]} x_{\gamma,t} + l_\gamma^2}, \quad , \forall \boldsymbol{\gamma \in \Gamma_n, n \in N_{flex}}. \tag{4.37}$$

4.6.1.5 Storage Facility

Set N_{bat} contains storage facilities. A battery is characterized by the charging and discharging efficiency parameters e_γ^c and e_γ^d, respectively, a maximum battery capacity $\overline{SOC_\gamma}$, a maximum power rate $\bar{x}_\gamma$ and a maximum number b_γ of full discharge cycles allowed. The set of control variables is $\mathcal{Y}_{n:n\in N_{\text{bat}}} = \{x_{\gamma,t}^{ch}, x_{\gamma,t}^{dis}, u_{\gamma,t}, SOC_{\gamma,t}\}$, where for timeslot t, variable $x_{\gamma,t}^{ch}$ is the charge power, $x_{\gamma,t}^{dis}$ is the discharge power, $u_{\gamma,t}$ is a binary variable denoting whether γ charges or discharges, and $SOC_{\gamma,t}$ is the battery's state of charge. A storage facility does not have an operational cost for DR, i.e.,

$$\boldsymbol{d_{\gamma,n}(\mathcal{Y}_n) = 0}, \quad \forall \boldsymbol{\gamma \in \Gamma_n, n \in N_{bat}} \tag{4.38}$$

but the operation of a battery is subject to the following set of constraints [68]:

$$
\begin{aligned}
&0 \le x_{\gamma,t}^{ch} \le u_{\gamma,t}\overline{\mathrm{x}}_{\gamma} \\
&0 \le x_{\gamma,t}^{dis} \le (1-u_{\gamma,t})\overline{\mathrm{x}}_{\gamma} \\
&SOC_{\gamma,t} = SOC_{\gamma,t-1} + \mathrm{e}_{\gamma}^{c} x_{\gamma,t}^{ch} - x_{\gamma,t}^{dis}/\mathrm{e}_{\gamma}^{d} \\
&0 \le SOC_{\gamma,t} \le \overline{\mathrm{SOC}}_{\gamma} \\
&SOC_{\gamma,|T|} \ge SOC_{\gamma,0} \\
&\textstyle\sum_{t\in T} x_{\gamma,t}^{dis} \le \mathrm{b}_{\gamma} \cdot \overline{\mathrm{SOC}}_{\gamma}.
\end{aligned}
\qquad (4.39) – (4.44)
$$

4.6.1.6 Facility with thermostatically controlled loads

Let N_{tcl} denote the set of facilities that feature thermostatically controlled loads (TCLs). Such facilities control the power consumption $x_{\gamma,t}$ of a TCL as well as indirectly controlling the room temperature $H_{\gamma,t}$, i.e. $\mathcal{Y}_{n:n\in N_{\mathrm{tcl}}} = \{x_{\gamma,t}, H_{\gamma,t}\}$. A TCL is characterized by minimum and a maximum acceptable temperature levels, denoted as $\underline{\mathrm{H}}_{\gamma,t}$ and $\overline{\mathrm{H}}_{\gamma,t}$ respectively. The TCL's temperature must be within $\left[\underline{\mathrm{H}}_{\gamma,t}, \overline{\mathrm{H}}_{\gamma,t}\right]$ at all times:

$$
\underline{\mathbf{H}}_{\boldsymbol{\gamma,t}} \le \boldsymbol{H_{\gamma,t}} \le \overline{\mathbf{H}}_{\boldsymbol{\gamma,t}}, \quad \forall \boldsymbol{t} \in \boldsymbol{T}, \boldsymbol{\gamma} \in \boldsymbol{\Gamma_n}, \boldsymbol{n} \in \boldsymbol{N_{\mathrm{tcl}}}. \qquad (4.45)
$$

The temperature transition depends on technical parameters h_{γ}^{ins}, h_{γ}^{eff} of the TCL that relate to the room's insulation and the TCL's efficiency, as well as on the TCL's initial temperature $\mathrm{H}_{\gamma,0}$ and the outdoors temperature H_t^{out}, as in:

$$
\begin{aligned}
H_{\gamma,t} = (1-\mathrm{h}_{\gamma}^{1})^{t}\mathrm{H}_{\gamma,0} - \sum_{\tau=1}^{t}(1-\mathrm{h}_{\gamma}^{\mathrm{ins}})^{(t-\tau)}\mathrm{H}_{t}^{\mathrm{out}} + \\
\sum_{\tau=1}^{t}(1-\mathrm{h}_{\gamma}^{\mathrm{ins}})^{(t-\tau)}\mathrm{h}_{\gamma}^{\mathrm{eff}} x_{\gamma,t}
\end{aligned}
\qquad (4.46)
$$

The DR cost function of a TCL is defined as the distance from its desired setpoint temperature $\widetilde{\mathrm{H}}_{\gamma,t}$ [69], i.e.:

$$
d_{\gamma,n}(\mathcal{Y}_n) = \mathrm{h}_{\gamma}^{\mathrm{cost}}(\widetilde{\mathrm{H}}_{\gamma,t} - H_{\gamma,t})^2, \quad \forall n \in N_{\mathrm{tcl}}. \qquad (4.47)
$$

4.6.1.7 Electric vehicles' charging station

An EV charging station $n \in N_{\mathrm{cs}}$ can form a flexible facility by scheduling the power consumption of its charging tasks. An EV $\gamma \in \Gamma_{n:n\in N_{\mathrm{cs}}}$ is characterized by an arrival time $\mathrm{e}_{\gamma}^{\mathrm{arr}}$, an energy requirement E_{γ}, a charging efficiency parameter $\mathrm{e}_{\gamma}^{\mathrm{eff}}$ and a maximum charging rate $\overline{\mathrm{x}}_{\gamma}$. The set of variables is $\mathcal{Y}_{n:n\in N_{\mathrm{cs}}} = \{x_{\gamma,t}, SOE_{\gamma,t}, u_{\gamma,t}\}$ where $SOE_{\gamma,t}$ is the state of energy in the EV's battery and $u_{\gamma,t}$ is a binary variable denoting whether the EV's charging demand has been satisfied in timeslot t. The set of constraints describing the EV model is:

$$x_{\gamma,t} = 0, \quad t < e_{\gamma}^{arr}$$
$$e_{\gamma}^{eff} \sum_{t \in T} x_{\gamma,t} = E_{\gamma}$$
$$SOE_{\gamma,t} = SOE_{\gamma,t-1} e_{\gamma}^{eff} x_{\gamma,t} \qquad (4.48) – (4.51)$$
$$u_{\gamma,t} = \begin{cases} 1, & SOE_{\gamma,t} - E_{\gamma} < 0 \\ 0, & SOE_{\gamma,t} - E_{\gamma} \geq 0 \end{cases}$$

Then, the DR cost is defined based on the extra waiting time that an EV suffers due to delayed charging (beyond its earliest possible task completion time $\lceil E_{\gamma}/e_{\gamma}^{eff}\bar{x}_{\gamma} \rceil$):

$$d_{\gamma,n}(y_n) = \sum_{t \in T} u_{\gamma,t} \cdot t - \lceil E_{\gamma}/e_{\gamma}^{eff}\bar{x}_{\gamma} \rceil - e_{\gamma}^{arr},$$
$$\forall \gamma \in \Gamma_n, n \in N_{cs}.$$

This formulation was first proposed in [70].

4.6.2 DSO network

We consider a 15-node radial distribution network (see figure below). The data for branches and loads are presented in the table below, adopted by [68]. The upper and lower bounds of the nodal voltage amplitude are set to 1.05 pu and 0.95 pu, respectively. We assume that 2 PV generators are installed at nodes 2 and 13 of the network, while 4 wind turbines are located at nodes 5, 8, 10 and 11. Their production curves are derived from . The base power and voltage are 1 MVA and 11kV. The cost c_a^{curt} of shedding 1 MW of RES generation was set to 50. For the evaluation, we considered a number of 100 assets for each facility.

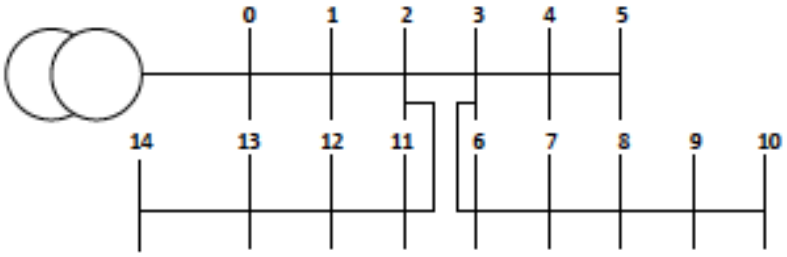

Figure 24: A 15-node radial distribution network

Table 6: Technical characteristics of the 15-node radial distribution network

Branch (#)	From Node (i)	To Node (a)	R_{ia} (pu)	X_{ia} (pu)	$\overline{P}_{ia}, \overline{Q}_{ia}$ (pu)	$\underline{P}_{ia}, \underline{Q}_{ia}$ (pu)	$\overline{V}_{ia}$ (pu)	$\underline{V}_{ia}$ (pu)
1	0	1	0.0031	0.0752	0.233	-0.233	0.95	1.05
2	1	2	0.0033	0.0018	0.233	-0.233	0.95	1.05
3	2	3	0.0067	0.0308	0.233	-0.233	0.95	1.05
4	3	4	0.0058	0.0149	0.233	-0.233	0.95	1.05
5	4	5	0.0141	0.0365	0.233	-0.233	0.95	1.05
6	6	6	0.0080	0.0369	0.233	-0.233	0.95	1.05
7	6	7	0.0090	0.0415	0.233	-0.233	0.95	1.05
8	7	8	0.0070	0.0323	0.233	-0.233	0.95	1.05
9	8	9	0.0037	0.0169	0.233	-0.233	0.95	1.05
10	9	10	0.0090	0.0415	0.233	-0.233	0.95	1.05
11	2	11	0.0275	0.1270	0.233	-0.233	0.95	1.05
12	11	12	0.0315	0.0814	0.233	-0.233	0.95	1.05
13	12	13	0.0396	0.1029	0.233	-0.233	0.95	1.05
14	13	14	0.0106	0.0041	0.233	-0.233	0.95	1.05

4.6.3 Cloud computing infrastructure

In our simulation experiments, we considered two topologies for the COMNET infrastructure with different characteristics in terms of the number of available resources and link lengths: a basic and an extended edge-fog-cloud topology (see the two figures below). For both network topologies, we assumed that the network is split into three layers, with Layer 1 representing edge, Layer 2 fog, and Layer 3 cloud infrastructure. The link lengths of the basic topology vary from 100 km to 500 km, whereas the extended topology features average link lengths of 150 km that vary on the interval of 30-500 km. The number, processing capacity and availability of the resources increase as we move to higher layers of the infrastructure (i.e. deeper in the cloud). We assume uniform processing capabilities at each node of a given layer. For the bottom layer, the processing capacity of a node was set to 9 GIPS (billion instructions per second). On the other hand, the utilization cost of the processing resources decreases from the edge to the cloud. The nodes of the different layers are interconnected through links of varying rates. Edge nodes are connected via lower rate links, while cloud nodes via higher speed links. However, in our performed simulation experiments the transmission latency was assumed to be negligible, given the small size of data that need to be transferred, and only the propagation latency was taken into consideration.

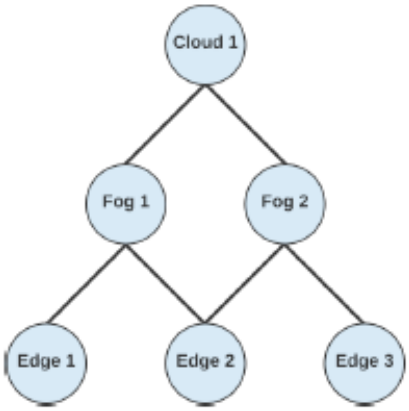

Figure 25: Basic network topology split into 3 layers to represent an edge-fog-cloud infrastructure

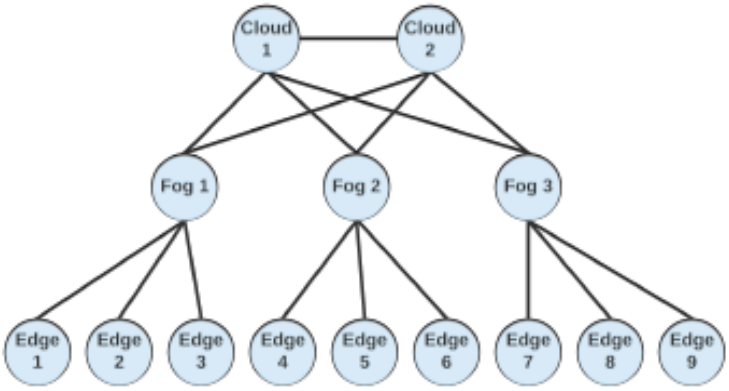

Figure 26: Extended network topology split into 3 layers to represent an edge-fog-cloud infrastructure

We examined the performance of the proposed ILP and heuristic assuming an instance of the DROaaS problem, in which a varying number of DR requests [10-60] need to be served. Each DR request refers to a certain set of facilities, the number of which was selected randomly from [2, 8]. We assumed that higher layer resources decrease the execution time of a task by 20% and the cost of utilizing processing power by 40% (1 c.u. for using layer 1 for 10 sec), in relation to lower layer resources (cloud-fog and fog-edge). The processing capacity of each node of the edge/fog/cloud layer was set to 9/10.8/12.96 GIPS respectively, while the respective cost of using resources for 10 sec was set to 1/1.4/1.96.

4.7 Performance evaluation results

Simulation experiments were performed, evaluating different scenarios in relation to the number of tasks, their processing and data requirements, the capacities of the computing and networking resources and their related costs. The computational load of line 4 of Algorithm 1, i.e. for solving the optimization problems (4.15) and (4.14) for each different facility type, were tested via simulations. The results are presented in the table below.

Table 7: Number of instructions of line 4 of Algorithm 1, for the DSO and each facility type

Facility type	Instructions count (in Millions)
DSO	16703
Curtailable loads	19369
Curtailable loads with ramps	874284
Time-shiftable loads	9028
Flexible loads	9026
Storage	15895
Thermostatically Controlled Loads	196891
EV charging station	120714

An interesting observation is that the computational cost for curtailable loads is massively increased by the sole introduction of ramp constraints. Moreover, the network loads, which relate to the volume of data necessary (number of parameters) to perform the calculations, are presented in the table below for each facility type. Finally, we should note that obtaining the optimal solution to problem (4.18) – (4.27) takes a prohibitive amount of time (in the order of hours). In contrast, the computational time of the Heuristic algorithm is only in the order of seconds, even for highly complex COMNET infrastructures. This makes the Heuristic algorithm applicable for the purposes of real-time electricity markets, which are typically cleared every 5 to 15 minutes. In what follows, we present simulation experiments that record the optimality loss of the fast Heuristic algorithm, compared to the optimal, but impractical, ILP. All simulation experiments were performed on a computer with an Intel Core i7-9700K processor running at 3,6 GHz and 32 GB of RAM. The simulations were run in Matlab using the CPLEX LP/MIP solver.

Table 8: Network load (in number of parameters needed to be communicated) for the DSO and each facility type

Facility type	Input (bytes)	Output (bytes)
Curtailable loads	5000	2400
Curtailable loads with ramps	5400	2400
Time-shiftable loads	700	2400
Flexible loads	800	2400
Storage	600	4800
Thermostatically Controlled Loads	7924	2400
EV charging station	600	2400

4.7.1 Results for the basic network topology

Initially, we evaluated the performance of the proposed ILP and heuristic mechanisms in relation to the total cost required to complete the execution of an iteration of the DR requests (see figure below). As expected, lower cost is achieved when the objective is set to minimize the processing per iteration cost (i.e. $w = 1$). In that case, the cloud resource nodes are preferred compared to the edge and fog nodes due to their lower cost and higher processing capabilities. The performance of the proposed ILP and heuristic is similar for a small number of DR requests, while for a higher number of DR requests, the ILP outperforms the heuristic. When the objective is the minimization of the latency for serving DR requests ($w = 0$), then more edge resources are utilized, resulting in increased cost when the heuristic mechanism is used. For these experiments, the latency bound for each DR request was set to 1.3 times the processing time of the largest task on the slowest resource.

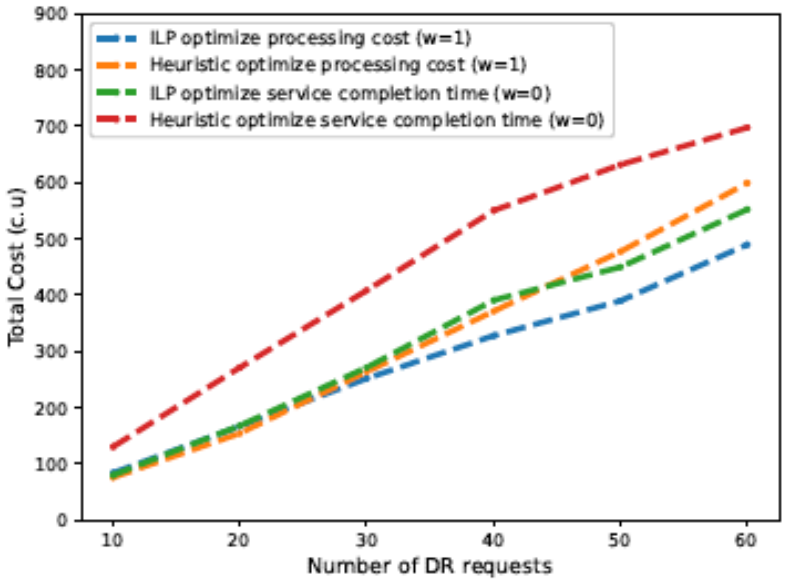

Figure 27: Total cost required to complete the execution of an iteration of the DR requests for the basic network topology

Next, we compared the total time required to complete an iteration (makespan) (see figure below) for the two developed mechanisms. In this case, the latency bound is relaxed. The best performance is achieved by the ILP with the objective of minimizing the latency for serving the DR requests, followed by the respective heuristic. The difference between the heuristic and the ILP is due to the fact that the ILP achieves the optimal allocation of the processing resources, while the heuristic with a worse performance in resource utilization, selects processing instances with slower computational capabilities to meet the objective criteria.

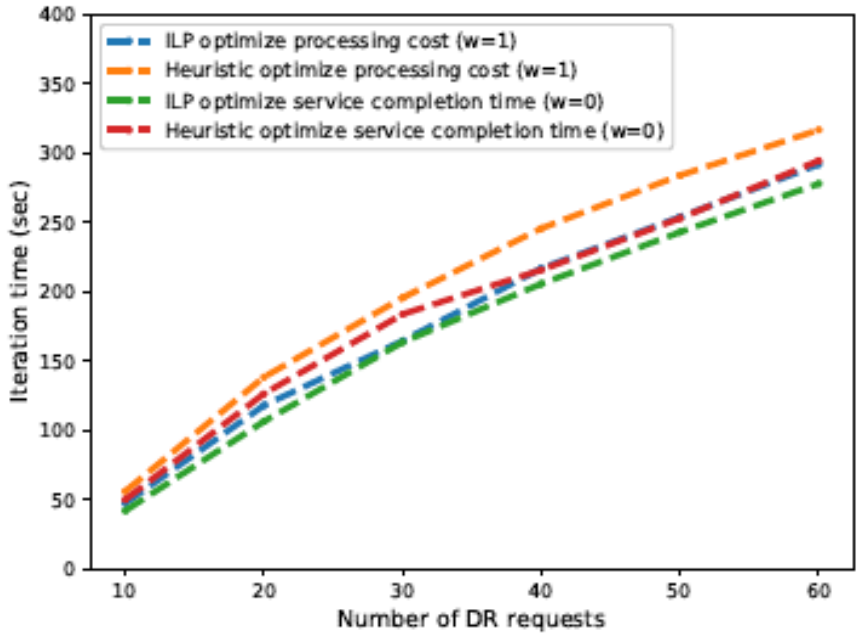

Figure 28: Total time required to complete an iteration for the ILP and the heuristic mechanism

4.7.2 Results for the extended network topology

We performed a number of experiments for the extended network topology, using the heuristic algorithm for two different cases. In the first case, we assumed a higher number of DR requests that vary from 200 to 1200, while the rest of our assumptions remained the same as in the basic network topology. In this case, we examined the allocation of resources at the different layers under the two objectives.

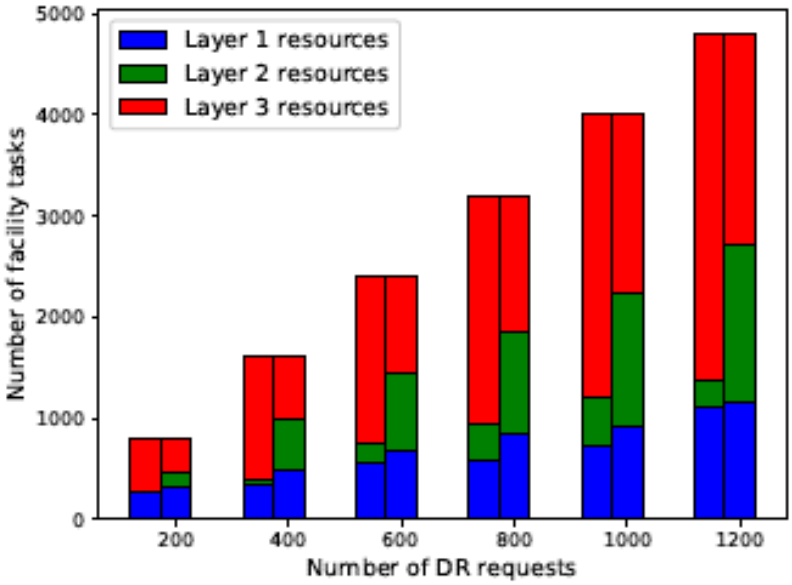

Figure 29: Resource allocation at the different network layers for the two objectives ($w = 1$ for the left bar, and $w = 0$ for the right bar) and the heuristic algorithm

As shown in the figure above, when the objective is the minimization of the processing cost (left bar), more tasks are executed in Layer 3. Moreover, the number of tasks executed in Layer 3 increases as the total number of DR request increase, taking advantage of the higher number and more powerful computational resources that are available in the cloud. In this case, the utilization of the Layer 2 resources is low. On the other hand, when the objective is the minimization of the service completion time (right bar), Layer 1 and Layer 2 resources are preferred. Especially the latter ones are highly utilized because of their advantage in accommodating the tasks that cannot be served by the Layer 1 resources due to the high waiting time that would violate the latency constraint. **Hence, when the main optimization criterion is the cost, the cloud resources are the most appropriate ones, but when the objective is the minimization of the service time, edge and fog resources are preferred as they offer shorter delays at the expense of a higher cost.**

In the second case examined, we assumed the use of special purpose hardware accelerators, such as GPUs, in the edge layer that provide a performance boost of 30% for the execution of DR facility tasks, compared to the general-purpose resources present in the fog and cloud layers. When the more powerful equipment is present at the edge, the edge resources are preferred under both objectives and tend to achieve significantly better performance compared to the case with our initial assumptions (see figure below). This is because the enhanced edge devices complete the tasks faster, as is depicted in both the processing cost and the total time required to complete an iteration. On the other hand, when no enhanced equipment is used at the edge, the completion time lags behind by 23% and 27% for the completion time and processing cost objectives, respectively.

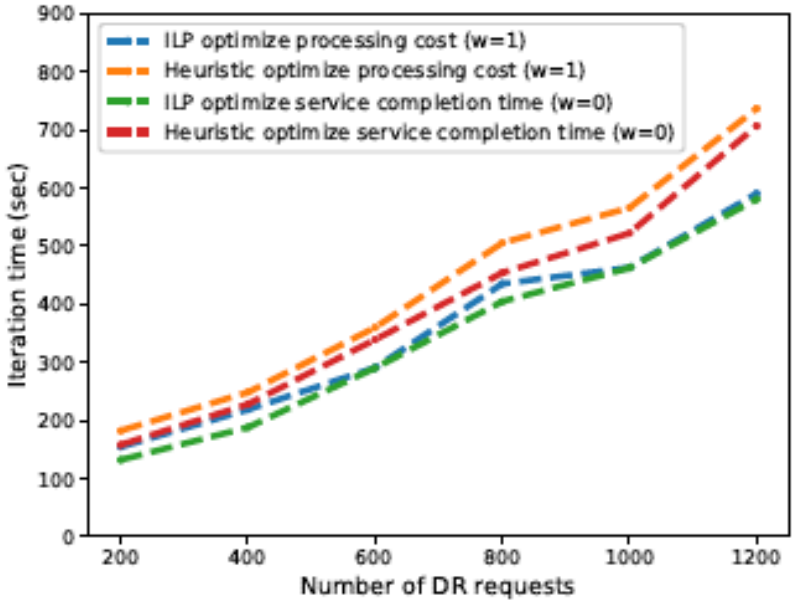

Figure 30: Total time required to complete an iteration for the ILP and the heuristic for the extended network with enhanced edge resources

4.8 Concluding remarks and lessons learned

Within FLEXGRID UCS 4.2 context, we develop advanced retail market mechanisms (ARMM) that can be used by an aggregator in order to operate a novel B2C flexibility market architecture. In this B2C flexibility market, various types of small-scale Distributed FlexAssets (DFAs) compete with each other, while the distribution network constraints are also taken into consideration (i.e. network-aware market clearing).

Following up the research results from the previous FLEXGRID D3.2, in this deliverable (D3.3), we focused on the **scalability and algorithmic complexity** of the proposed B2C flexibility market. More specifically, we considered a large number of FlexRequests published by the FLEXGRID ATP, a large portfolio of end users (i.e. at a scale of several hundreds of end users or even millions), more complex (and thus realistic) FlexAsset models and more stringent real-time constraints imposed by the B2C flexibility market.

Thus, we modeled, formulated and provided performance evaluation results for the problem of clearing a B2C flexibility market of a power distribution network using a diverse set of computational resources (edge, fog, cloud) over the cloud continuum. We presented a B2C flexibility market clearing algorithm based on Lagrangian relaxation, and we configured the algorithm's execution with a computational resource allocation algorithm. For the cloud resource allocation, we presented an **optimal algorithm and a heuristic that achieves near-optimal performance while dramatically reducing the computational time**. The resource allocation mechanism is able to service multiple B2C flexibility markets (i.e. for many distribution network areas at the same time), and leverage an economy-of-scale effect towards minimizing the cost of computational resources, while respecting the execution time constraints of each FlexRequest pubished by the DSOs.

Our experimental results demonstrate the effect of different FlexAsset models in the resulting computational burden (e.g. the sole introduction of ramp constraints had a dramatic effect), the trade-off between optimality and scalability (as approached by the optimal solution and a faster but sub-optimal heuristic), as well as the resulting allocation of computational tasks through the different layers (edge / fog / cloud) of the envisaged architecture. The heuristic algorithm manages to efficiently address B2C flexibility markets of different size and complexity, with its performance depending on the processing capacity and availability of edge/fog, and secondarily cloud, processing resources, since the communication requirements are minimal.

Conclusively, our work enables a new business model, where an independent aggregator can offer the functionality of DR operation as a service (DROaaS) by optimally exploiting a cloud/fog/edge infrastructure. Thus, the aggregator will be able to manage a large number of concurrent FlexRequests being published by any possible "FlexBuyer" entity and execute respective network-aware market clearing processes in order to automatically aggregate the required flexibility from the end users/consumers with the least possible cost.

After communicating FLEXGRID UCS 4.2 scientific results to both academic and industrial communities, we have come up with a short list of lessons learned that could be further investigated in future R&I initiatives. The table below summarizes research and business-related insights for each one of the lessons learned. The most interesting aspect is that the cloud-fog-edge computing continuum can be exploited in several Generic Business Processes (GBPs[11]) that pose respective generic business requirements (i.e. computational/ modeling complexity, need for parallelization of vast amount of computing tasks, real-time responsiveness, scalability and interoperability and security/privacy/trust requirements) to the smart grid business ecosystem. More details about the five main GBPs that could be facilitated via the use of the proposed cloud-fog-edge computing continuum can be seen in the table below.

Lesson learned	Research & Business insights
Need for deep inter-disciplinary research between power engineers and ICT engineers in order to exploit the cloud/edge computing paradigm for many smart grid applications. Many complex mathematical models, which are decomposable could be split into many smaller problems and thus run in parallel in multiple H/W processing resources (i.e. computers).	Development of an integrated and holistic energy system able to combine orthogonal technologies (e.g. DR concept from the energy sector and cloud/edge computing from the ICT sector) in order to offer an attractive trade-off between cost-efficiency and scalability, while also preserving the participants' (i.e. end users') privacy. Similarly to the proposed B2C flexibility market clearing solution, today's EU

[11] GBP is a term used by H2020 BRIDGE initiative in order to categorize some general business processes, which are related to flexibility use in smart grids - https://ec.europa.eu/energy/sites/default/files/documents/bridge_wg_data_management_interoperability_of_flexibility_assets_report_2020-2021.pdf

	Market Operators (like Nord Pool and NODES) could also exploit FLEXGRID research work to parallelize the existing market clearing processes and thus achieve much less execution times and facilitate much more complex modelling, which is a major business requirement towards achieving optimal market efficiency.
GBP class #1: Too complex/intractable model (and algorithm) that cannot be solved or needs several days/months to run in a powerful central server of a given market actor (e.g. System Operator). By parallelizing this type of algorithm and running it distributedly in the edge-cloud computing continuum, it can actually be solved.	Applicability in real business cases: i) Co-optimization of FlexAsset investments between a System Operator and profit-based Energy Service Providers (ESPs) to minimize network upgrade investments, ii) system flexibility planning, iii) RES and Flexibility Asset planning/investment problem, iv) other Equilibrium Problems with Equilibrium Constraints (EPEC) trying to model complex interactions among several energy market stakeholders (e.g. TSO, DSO, ESP, end users).
GBP class #2: Too many processes and/or "what-if" scenarios that should be run within one hour or less (e.g. many scenarios in a stochastic optimization or risk management model, too many network zones/nodes in a market clearing model). This class of problems usually refers to day-ahead problems that are very often for several key stakeholders in the energy sector. By running these GBPs distributedly in the edge-cloud computing continuum, we can considerably decrease the execution time of these GBPs.	Applicability in real business cases: i) day-ahead market clearing, ii) day-ahead forecasting, iii) day-ahead scheduling, etc.
GBP class #3: The model/algorithm is relatively not so complex, but there are stringent real-time constraints that should be satisfied. By running it distributedly in the edge-cloud computing continuum, we can guarantee a solution within the stringent time constraints and considerably improve end-to-end response times.	Applicability in real business cases: i) balancing market clearing, ii) near-real-time congestion management, iii) intra-day operational challenges such as near-real-time forecasting, near-real-time flexibility offers by aggregator/ESP, etc.
GBP class #4: The model/algorithm needs to be scalable in order to run for many more physical assets/entities and/or with much more data, which are mainly produced at the edge. In this context, communication load requirements will be considerably increased, so joint management	Applicability in real business cases: i) AI training and inference at the edge, ii) distributed machine learning models, iii) uncertainty management by using big data and AI models such as Deep Neural Networks (DNNs), etc.

of both computing and networking resources is required. By running it distributedly, we can exploit the virtually infinite capacity of the edge-cloud computing continuum and thus achieve good trade-off solution between results' accuracy and modeling complexity.	
GBP class #5: Data/knowledge sharing model and respective algorithmic solution for the efficient interaction between two or more energy market stakeholders. It refers to an iterative process in which each actor runs an instance of the algorithmic process in each own computing infrastructure (edge) and communicates only some market price signals (e.g. Lagrange mutlipliers, KKTs, etc.) to another actor that uses these signals as input to compute its own problem in its own computing infrastructure. Thus, there is no need to communicate private/sensitive information boosting thus business interactions within the smart grid ecosystem (cf. trust requirement) without compromising privacy and security-related requirements.	Applicability in real business cases: Deal with the need for cyber-security, privacy and trust among involved stakeholders, who want to cooperate with each other in order to realize "win-win" business contexts, but they cannot/are not willing to share sensitive information about their network topology, operational data, business strategy, etc.

5 Conclusions

The aggregator is an emerging role in the developing electricity market. Within WP3 of FLEXGRID, the focus of research was on the development of tools to enable and facilitate the operation and orchestration of an aggregated flexibility portfolio. Different approaches and use case scenarios were investigated reflecting different types of interactions with the market (flexibility buyers) and end-users (flexibility providers) and market designs. Concluding remarks, lessons learned and research and business insights for each approach are summarized in the final sections of chapters 2, 3 and 4.

The research work of WP3 is now being integrated within the AFAT and FLEXGRID ATP (WP6). More specifically, stable versions of the algorithms of UCS 4.1 – "Manage a FlexRequest", UCS 4.2 – "Manage a B2C flexibility market" and UCS 4.3 – "Create a FlexOffer" are being integrated within the ongoing work of WP6. These tools will facilitate the aggregator actor towards the operation and management of its portfolio and allow both online operation for real-time support and offline operation for "what-if" simulations and testing under different business scenarios.

Aggregator (AFAT) services are also being validated in pilot sites within the ongoing work of WP7. Two methods of WP3, namely "Manage a FlexRequest" and "Create a FlexOffer" (UCS 4.1 and UCS 4.3 respectively), are being used for optimal aggregation of flexibility of pilot assets, with an extension of virtual assets for different business cases.

The research results of all three use case scenarios of WP3 are Key Exploitable Results (KERs) of the FLEXGRID project and the research outcome will be used for the final development and enhancement of the business models and value proposition regarding the aggregator.

In the figure below, the timeline schedule of WP3 is illustrated. Milestone #9 has been achieved with this deliverable, which concludes all milestones of WP3.

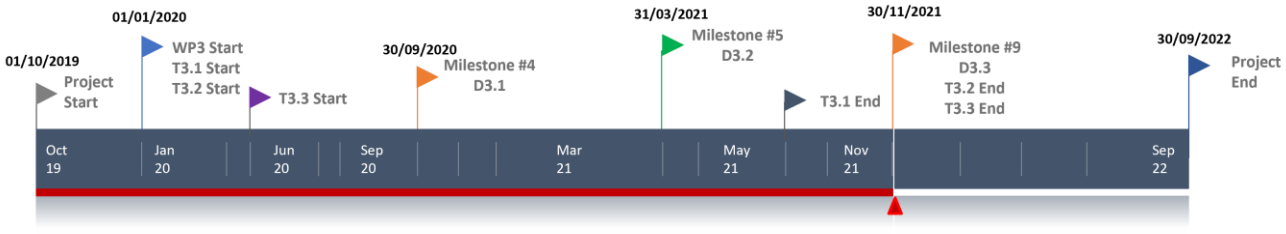

Figure 31: FLEXGRID project's and WP3 timeline schedule (All milestones have been accomplished)

References

1 I. Mamounakis, N. Efthymiopoulos, P. Makris, D. J. Vergados, G. Tsaousoglou, and E. M. Varvarigos, “A novel pricing scheme for managing virtual energy communities and promoting behavioral change towards energy efficiency,” Electric Power Systems Research, vol. 167, pp. 130 – 137, 2019.

2 K. De Craemer, S. Vandael, B. Claessens, and G. Deconinck, “An event-driven dual coordination mechanism for demand side management of phevs,” IEEE Transactions on Smart Grid, vol. 5, no. 2, pp. 751–760, 2013.

3 G. Tsaousoglou, J. S. Giraldo, P. Pinson, and N. G. Paterakis, “Mechanism design for fair and efficient dso flexibility markets,” IEEE Transactions on Smart Grid, pp. 1–1, 2020.

4 K. Seklos, G. Tsaousoglou, K. Steriotis, N. Efthymiopoulos, P. Makris, and E. Varvarigos, “Designing a distribution level flexibility market using mechanism design and optimal power flow,” in 2020 International Conference on Smart Energy Systems and Technologies (SEST), 2020, pp. 1–6.

5 G. Tsaousoglou, K. Steriotis, N. Efthymiopoulos, K. Smpoukis, and E. Varvarigos, “Near-optimal demand side management for retail electricity markets with strategic users and coupling constraints,” Sustainable Energy, Grids and Networks, vol. 19, p. 100236, 2019.

6 C. P. Mediwaththe, M. Shaw, S. Halgamuge, D. B. Smith, and P. Scott, “An incentive-compatible energy trading framework for neighborhood area networks with shared energy storage,” IEEE Transactions on Sustainable Energy, vol. 11, no. 1, pp. 467–476, 2019.

7 G. Tsaousoglou, K. Steriotis, N. Efthymiopoulos, P. Makris, and E. Varvarigos, “Truthful, Practical and Privacy-aware Demand Response in the Smart Grid via a Distributed and Optimal Mechanism,” IEEE Transactions on Smart Grid, pp. 1–1, 2020.

8 G. Tsaousoglou, P. Pinson, and N. G. Paterakis, “Transactive energy for flexible prosumers using algorithmic game theory,” IEEE Transactions on Sustainable Energy, pp. 1–1, 2021.

9 Y. Zhang, N. Rahbari-Asr, J. Duan, and M.-Y. Chow, “Day-ahead smart grid cooperative distributed energy scheduling with renewable and storage integration,” IEEE Transactions on Sustainable Energy, vol. 7, no. 4, pp. 1739–1748, 2016.

10 W. Tang, S. Bi, and Y. J. Zhang, “Online charging scheduling algorithms of electric vehicles in smart grid: An overview,” IEEE communications Magazine, vol. 54, no. 12, pp. 76–83, 2016.

11 R. Deng, Z. Yang, J. Chen, N. R. Asr, and M. Chow, “Residential energy consumption scheduling: A coupled-constraint game approach,” IEEE Transactions on Smart Grid, vol. 5, no. 3, pp. 1340–1350, 2014.

12 S. Chakraborty, M. Cvetkovic, K. Baker, R. Verzijlbergh, and Z. Lukszo, “Consumer hedging against price volatility under uncertainty,” in 2019 IEEE Milan PowerTech, 2019, pp. 1–6.

13 J. Sachs and O. Sawodny, “A two-stage model predictive control strategy for economic diesel-pv-battery island microgrid operation in rural areas,” IEEE Transactions on Sustainable Energy, vol. 7, no. 3, pp. 903–913, 2016.

14 V. Lakshminarayanan, V. G. S. Chemudupati, S. K. Pramanick, and K. Rajashekara, “Real-time optimal energy management controller for electric vehicle integration in workplace microgrid,” IEEE Transactions on Transportation Electrification, vol. 5, no. 1, pp. 174–185, 2019.

15 J. Zhu, Z. Yang, Y. Chang, Y. Guo, K. Zhu, and J. Zhang, “A novel lstm based deep learning approach for multi-time scale electric vehicles charging load prediction,” in 2019 IEEE Innovative Smart Grid Technologies - Asia (ISGT Asia), 2019, pp. 3531–3536.
16 S. Minniti, A. N. M. M. Haque, N. G. Paterakis, and P. H. Nguyen, “A hybrid robust-stochastic approach for the day-ahead scheduling of an ev aggregator,” in 2019 IEEE Milan PowerTech, 2019, pp. 1–6.
17 J. L. Crespo-Vazquez, T. AlSkaif, M. Gonzalez-Rueda, and M. Gibescu, “A community-based energy market design using decentralized decision making under uncertainty,” IEEE Transactions on Smart Grid, vol. 12, no. 2, pp. 1782–1793, 2021.
18 N. Neyestani, M. Yazdani-Damavandi, M. Shafie-Khah, G. Chicco, and J. P. Catalao, “Stochastic modeling of multi-energy carriers dependencies in smart local networks with distributed energy resources,” IEEE Transactions on Smart Grid, vol. 6, no. 4, pp. 1748–1762, 2015.
19 D. Thomas, O. Deblecker, and C. S. Ioakimidis, “Optimal operation of an energy management system for a grid-connected smart building considering photovoltaics’ uncertainty and stochastic electric vehicles’ driving schedule,” Applied Energy, vol. 210, pp. 1188 – 1206, 2018.
20 A. Papavasiliou, Y. Mou, L. Cambier, and D. Scieur, “Application of stochastic dual dynamic programming to the real-time dispatch of storage under renewable supply uncertainty,” IEEE Transactions on Sustainable Energy, vol. 9, no. 2, pp. 547–558, 2017.
21 R. Lu, T. Ding, B. Qin, J. Ma, X. Fang, and Z. Y. Dong, “Multistage stochastic programming to joint economic dispatch for energy and reserve with uncertain renewable energy,” IEEE Transactions on Sustainable Energy, 2019.
22 M. Majidi and K. Zare, “Integration of smart energy hubs in distribution networks under uncertainties and demand response concept,” IEEE Transactions on Power Systems, vol. 34, no. 1, pp. 566–574, 2018.
23 T. M. Hansen, E. K. P. Chong, S. Suryanarayanan, A. A. Maciejewski, and H. J. Siegel, “A partially observable markov decision process approach to residential home energy management,” IEEE Transactions on Smart Grid, vol. 9, no. 2, pp. 1271–1281, 2018.
24 Y. Yang, Q. Jia, G. Deconinck, X. Guan, Z. Qiu, and Z. Hu, “Distributed coordination of ev charging with renewable energy in a microgrid of buildings,” IEEE Transactions on Smart Grid, vol. 9, no. 6, pp. 6253–6264, 2018.
25 C. Keerthisinghe, G. Verbiˇc, and A. C. Chapman, “A fast technique for smart home management: Adp with temporal difference learning,” IEEE Transactions on Smart Grid, vol. 9, no. 4, pp. 3291–3303, 2018.
26 S. Vandael, B. Claessens, M. Hommelberg, T. Holvoet, and G. Deconinck, “A scalable three-step approach for demand side management of plug-in hybrid vehicles,” IEEE Transactions on Smart Grid, vol. 4, no. 2, pp. 720–728, 2013.
27 D. Li and S. K. Jayaweera, “Distributed smart-home decision-making in a hierarchical interactive smart grid architecture,” IEEE Transactions on Parallel and Distributed Systems, vol. 26, no. 1, pp. 75–84, 2015.
28 Z. Pan, T. Yu, J. Li, K. Qu, L. Chen, B. Yang, and W. Guo, “Stochastic transactive control for electric vehicle aggregators coordination: A decentralized approximate dynamic programming approach,” IEEE Transactions on Smart Grid, vol. 11, no. 5, pp. 4261–4277, 2020.

29 H. Sun, X. Chen, Q. Shi, M. Hong, X. Fu, and N. D. Sidiropoulos, “Learning to optimize: Training deep neural networks for interference management,” IEEE Transactions on Signal Processing, vol. 66, no. 20, pp. 5438–5453, 2018.
30 M. K. Singh, S. Gupta, V. Kekatos, G. Cavraro, and A. Bernstein, “Learning to optimize power distribution grids using sensitivity informed deep neural networks,” in 2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm). IEEE, 2020, pp. 1–6.
31 M. Shin, D. Choi, and J. Kim, “Cooperative management for pv/ess-enabled electric vehicle charging stations: A multiagent deep reinforcement learning approach,” IEEE Transactions on Industrial Informatics, vol. 16, no. 5, pp. 3493–3503, 2020.
32 Y. Du and F. Li, “Intelligent multi-microgrid energy management based on deep neural network and model-free reinforcement learning,” IEEE Transactions on Smart Grid, vol. 11, no. 2, pp. 1066–1076, 2020.
33 K. L. Lopez, C. Gagne, and M. Gardner, “Demand-side management using deep learning for smart charging of electric vehicles,” IEEE Transactions on Smart Grid, vol. 10, no. 3, pp. 2683–2691, 2019.
34 F. de Nijs, “Resource-constrained multi-agent markov decision processes,” PhD Dissertation, Delft University of Technology, 2019.
35 Z. Wan, H. Li, H. He, and D. Prokhorov, “Model-free real-time ev charging scheduling based on deep reinforcement learning,” IEEE Transactions on Smart Grid, vol. 10, no. 5, pp. 5246–5257, 2019.
36 H. Li, Z. Wan, and H. He, “Constrained ev charging scheduling based on safe deep reinforcement learning,” IEEE Transactions on Smart Grid, vol. 11, no. 3, pp. 2427–2439, 2020.
37 J. Achiam, D. Held, A. Tamar, and P. Abbeel, “Constrained policy optimization,” in Proc. 34th Int. Conf. Mach. Learn., vol. 70, no. 10, 2017, p. 22–31.
38 S. Bera, S. Misra, and J. J. Rodrigues, “Cloud computing applications for smart grid: A survey,” IEEE Transactions on Parallel and Distributed Systems, vol. 26, no. 5, pp. 1477–1494, 2014.
39 M. Yigit, V. C. Gungor, and S. Baktir, “Cloud computing for smart grid applications,” Computer Networks, vol. 70, pp. 312–329, 2014.
40 X. Fang, D. Yang, and G. Xue, “Evolving smart grid information management cloudward: A cloud optimization perspective,” IEEE Transactions on Smart Grid, vol. 4, no. 1, pp. 111–119, 2013.
41 E. Dall’Anese, K. Baker, and T. Summers, “Chance-constrained ac optimal power flow for distribution systems with renewables,” IEEE Transactions on Power Systems, vol. 32, no. 5, pp. 3427–3438, 2017.
42 T. Soares, R. J. Bessa, P. Pinson, and H. Morais, “Active distribution grid management based on robust ac optimal power flow,” IEEE Transactions on Smart Grid, vol. 9, no. 6, pp. 6229–6241, 2017.
43 M.-M. Zhao, Q. Shi, Y. Cai, M.-J. Zhao, and Y. Li, “Distributed penalty dual decomposition algorithm for optimal power flow in radial networks,” IEEE Transactions on Power Systems, vol. 35, no. 3, pp. 2176–2189, 2019.
44 X. Pan, A. Jiang, and H. Wang, “Edge-cloud computing application, architecture, and challenges in ubiquitous power internet of things demand response,” Journal of Renewable and Sustainable Energy, vol. 12, no. 6, p. 062702, 2020.

45 C. Feng, Y. Wang, Q. Chen, G. Strbac, and C. Kang, “Smart grid encounters edge computing: Opportunities and applications.” Advances in Applied Energy, p. 100006, 2020.
46 X. Zhang, D. Biagioni, M. Cai, P. Graf, and S. Rahman, “An edge-cloud integrated solution for buildings demand response using reinforcement learning,” IEEE Transactions on Smart Grid, vol. 12, no. 1, pp. 420–431, 2020.
47 M. H. Yaghmaee, A. Leon-Garcia, and M. Moghaddassian, “On the performance of distributed and cloud-based demand response in smart grid,” IEEE Transactions on Smart Grid, vol. 9, no. 5, pp. 5403–5417, 2017.
48 S. Chen, L. Jiao, L. Wang, and F. Liu, “An online market mechanism for edge emergency demand response via cloudlet control,” in IEEE INFOCOM 2019-IEEE Conference on Computer Communications. IEEE, 2019, pp. 2566–2574.
49 N. Kulkarni, S. Lalitha, and S. A. Deokar, “Real time control and monitoring of grid power systems using cloud computing.” International Journal of Electrical & Computer Engineering (2088-8708), vol. 9, no. 2, 2019.
50 K. Shahryari and A. Anvari-Moghaddam, “Demand side management using the internet of energy based on fog and cloud computing,” in 2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData). IEEE, 2017, pp. 931–936.
51 A. Jiang, H. Wei, J. Deng, and H. Qin, “Cloud-edge cooperative model and closed-loop control strategy for the price response of large-scale air conditioners considering data packet dropouts,” IEEE Transactions on Smart Grid, vol. 11, no. 5, pp. 4201–4211, 2020.
52 L. Ruan, Y. Yan, S. Guo, F. Wen, and X. Qiu, “Priority-based residential energy management with collaborative edge and cloud computing,” IEEE Transactions on Industrial Informatics, vol. 16, no. 3, pp. 1848–1857, 2019.
53 M. Dabbaghjamanesh, A. Kavousi-Fard, and Z. Y. Dong, “A novel distributed cloud-fog based framework for energy management of networked microgrids,” IEEE Transactions on Power Systems, vol. 35, no. 4, pp. 2847–2862, 2020.
54 M. H. Y. Moghaddam and A. Leon-Garcia, “A fog-based internet of energy architecture for transactive energy management systems,” IEEE Internet of Things Journal, vol. 5, no. 2, pp. 1055–1069, 2018.
55 Y. Li, Y. Mao, X. Zeng, D. Liu, Y. Zu, and L. Mei, “A novel energy trading platform for distribution network based on edge computing,” in 2019 IEEE 3rd Conference on Energy Internet and Energy System Integration (EI2). IEEE, 2019, pp. 2625–2629.
56 K. Kaur, S. Garg, G. Kaddoum, S. H. Ahmed, F. Gagnon, and M. Atiquzzaman, “Demand-response management using a fleet of electric vehicles: An opportunistic-sdn-based edge-cloud framework for smart grids,” IEEE Network, vol. 33, no. 5, pp. 46–53, 2019.
57 Y. Shang, M. Liu, Z. Shao, and L. Jian, “Internet of smart charging points with photovoltaic integration: A high-efficiency scheme enabling optimal dispatching between electric vehicles and power grids,” Applied Energy, vol. 278, p. 115640, 2020.
58 M. A. Al Faruque and K. Vatanparvar, “Energy management-as-a-service over fog computing platform,” IEEE internet of things journal, vol. 3, no. 2, pp. 161–169, 2015.
59 M. E. Baran and F. F. Wu, “Network reconfiguration in distribution systems for loss reduction and load balancing,” IEEE Power Engineering Review, vol. 9, no. 4, pp. 101–102, 1989.

60 P. Samadi, A. Mohsenian-Rad, R. Schober, V. W. S. Wong, and J. Jatskevich, “Optimal real-time pricing algorithm based on utility maximization for smart grid,” in 2010 First IEEE International Conference on Smart Grid Communications, 2010, pp. 415–420.
61 G. Tsaousoglou, N. Efthymiopoulos, P. Makris, and E. Varvarigos, “Personalized real time pricing for efficient and fair demand response in energy cooperatives and highly competitive flexibility markets,” Journal of Modern Power Systems and Clean Energy, vol. 7, no. 1, pp. 151–162, 2019.
62 A. Mohsenian-Rad and A. Leon-Garcia, “Optimal residential load control with price prediction in real-time electricity pricing environments,” IEEE Transactions on Smart Grid, vol. 1, no. 2, pp. 120–133, 2010.
63 G. Tsaousoglou, K. Steriotis, N. Efthymiopoulos, K. Smpoukis, and E. Varvarigos, “Near-optimal demand side management for retail electricity markets with strategic users and coupling constraints,” Sustainable Energy, Grids and Networks, vol. 19, p. 100236, 2019.
64 P. Samadi, A. Mohsenian-Rad, R. Schober, V. W. S. Wong, and J. Jatskevich, “Optimal real-time pricing algorithm based on utility maximization for smart grid,” in 2010 First IEEE International Conference on Smart Grid Communications, 2010, pp. 415–420.
65 G. Tsaousoglou, N. Efthymiopoulos, P. Makris, and E. Varvarigos, “Personalized real time pricing for efficient and fair demand response in energy cooperatives and highly competitive flexibility markets,” Journal of Modern Power Systems and Clean Energy, vol. 7, no. 1, pp. 151–162, 2019.
66 A. Mohsenian-Rad and A. Leon-Garcia, “Optimal residential load control with price prediction in real-time electricity pricing environments,” IEEE Transactions on Smart Grid, vol. 1, no. 2, pp. 120–133, 2010.
67 G. Tsaousoglou, K. Steriotis, N. Efthymiopoulos, K. Smpoukis, and E. Varvarigos, “Near-optimal demand side management for retail electricity markets with strategic users and coupling constraints,” Sustainable Energy, Grids and Networks, vol. 19, p. 100236, 2019.
68 K. Steriotis, K. Smpoukis, N. Efthymiopoulos, G. Tsaousoglou, P. Makris, and E. M. Varvarigos, “Strategic and network-aware bidding policy for electric utilities through the optimal orchestration of a virtual and heterogeneous flexibility assets’ portfolio,” Electric Power Systems Research, vol. 184, p. 106302, 2020.
69 G. Tsaousoglou, K. Steriotis, N. Efthymiopoulos, P. Makris, and E. Varvarigos, “Truthful, practical and privacy-aware demand response in the smart grid via a distributed and optimal mechanism,” IEEE Transactions on Smart Grid, 2020.
70 G. Tsaousoglou, P. Pinson, and N. G. Paterakis, “Max-min fairness for demand side management under high res penetration: Dealing with undefined consumer valuation functions,” in 2020 International Conference on Smart Energy Systems and Technologies (SEST), 2020, pp. 1–6.
71 “Github of h2020 project vimsen,” https://github.com/vimsen/dss/blob/master/datapoints/oikiakoi.csv, 2015.