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

Intermediate Version of Distributed Flexibility Asset Markets and Advanced Retail Market Mechanisms

Deliverable D3.2

Document Information

Scheduled delivery	31.03.2021
Actual delivery	22.03.2021
Version	Final
Responsible Partner	ICCS

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
AFAT	Automated Flexibility Aggregation Toolkit
AI	Artificial Intelligence
API	Application Programming Interface
ARMM	Advanced Retail Market Mechanism
ATP	Automated Trading Platform
B2B/B2C	Business to Business / Business to Consumer
BIC	Bayes-Nash Incentive Compatibility
BRP	Balance Responsible Party
CA	Clinching Auction
DAM	Data Acquisition Module
DER	Distributed Energy Resource
DFA	Distributed Flexibility Asset
DNN	Deep Neural Network
DR	Demand Response
DSIC	Dominant-Strategy-Incentive-Compatibility
DSM	Demand Side Management
DSO/TSO	Distribution/Transmission System Operator
ECC	Energy Consumption Curve
ESP	Energy Service Provider
ESS	Energy Storage System
EV	Electric Vehicle
FSP	Flexibility Service Provider
FST	FlexSupplier's Toolkit
GUI	Graphical User Interface
HVAC	Heating, Ventilation and Air Conditioning
ICT	Information and Communication Technology
KPI	Key Performance Indicator
MCA	Modified Clinching Auction
MDP	Markov Decision Process
ML	Machine Learning
MM	Market Mechanism
MTU	Market Time Unit

RES	Renewable Energy Sources
RF	Random Forest
S/W	Software
TCL	Thermostatically Controlled Loads
TOU	Time of Use
VCG	Vickrey-Clarke-Groves
VPP	Virtual Power Plant
WEMM	Wholesale Electricity Market Module

Table of Contents

List of Figures and Tables

1.1 List of Figures

1.2 List of Tables

Document History

This deliverable includes the first 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
30/11/2020	v0.1	Draft ToC circulated within all consortium partners
07/01/2021	v0.2	All partners commented on the draft ToC structure.
08/01/2021	v0.3	Final ToC version has been agreed and writing task delegations have been provided to ICCS and UCY.
03/02/2021	v0.4	ICCS wrote chapters 3 and 4 and contributed its part in chapter 5.
26/02/2021	v0.6	UCY wrote chapter 2 and contributed its part in chapter 5.
08/03/2021	v0.8	ICCS integrated all contributions and pre-final D3.2 version has been sent to UNIZG-FER for internal review.
15/03/2021	v0.9	UNIZG-FER made a thorough review and requested for changes to enhance the quality of the deliverable.
20/03/2021	v0.95	ICCS and UCY addressed all comments from the internal review process and forwarded the final version to the coordinator.
22/03/2021	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 implemented in a S/W toolkit called Automated Flexibility Aggregation Toolkit (AFAT), which will dynamically interact with the core FLEXGRID Automated Trading Platform (ATP).

Chapter 1 brings an introduction to this report summarizing the scope and purpose of the document. More specifically, it provides a high-level description and summary of: i) the aggregator's business interests and how are these inter-related with the residual FLEXGRID business ecosystem, ii) state-of-the-art solutions for the aggregator's business challenges, iii) proposed research problems' statements, which are based on (i) and (ii), and what are the FLEXGRID's innovations, and iv) FLEXGRID's potential research impact on future aggregator's business.

Chapters 2-4 follow a similar structure in order to present the WP3 research results in a coherent manner. In particular, for each one of the three respective research problems, we present:

- Problem statement, related state-of-the-art and summary of FLEXGRID research contributions
- Proposed system model under study
- Problem formulation including all mathematical modeling
- Proposed algorithmic solution
- Simulation setup and performance evaluation results
- Next steps on how to elaborate on the ongoing WP3 research work until M26.

Chapter 2 presents the research problem of the FLEXGRID UCS 4.1 entitled "Aggregator manages a FlexRequest". Here, we assume that a flexibility market has been cleared and the aggregator needs to optimally schedule its portfolio. 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 the associated costs. The revenues of the aggregator increase with positive responses to FlexRequests. The associated costs can be divided into two categories. The first are end-user compensations for provision of flexibility, defined in FlexContracts. The second involves potential imbalance costs, meaning the financial effect of activating flexibility and deviating from the baseline (already scheduled energy profile of the flexibility assets due to their participation in the day-ahead energy market).

Chapter 3 presents the research problem of the FLEXGRID UCS 4.3 that can be summarized as "Aggregator creates a FlexOffer in an automated and dynamic way". Here, a novel bidding

algorithm is proposed in order for the aggregator to be able to participate in near-real-time balancing/flexibility markets. In more detail, we propose a generic method for capturing the aggregator's upward and downward flexibility cost for a set of distributed FlexAssets (or else DERs) that have non-convex models and inter-temporal couplings. The method is model-free in the sense that it is not tailored to any specific DER model. Rather, it can be applied to any type of FlexAssets, regardless of the specific model that each type of FlexAsset has. For this purpose, we use a fitting function and machine-learning (ML) methods to evaluate the performance of the proposed FLEXGRID intelligence.

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. The allocation rule refers to the way that the aggregator decides upon how much net 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 that 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.

Chapter 5 presents how all the above-mentioned research novelties that have been tested and validated at TRL 3, will be integrated in the Automated Flexibility Aggregation Toolkit (AFAT), which is part of the FLEXGRID Automated Trading Platform (ATP) at TRL 5. In particular, the AFAT’s frontend and backend services are described as well as the interaction between the WP3 research work and WP6 S/W implementation and integration work.

Conclusively, in Chapter 6, we summarize the next steps for WP3 research work. We also describe how the WP3 research results will be elaborated in other Work Packages until the end of the project’s lifetime.

1 Introduction

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

The purpose of High Level Use Case (HLUC) #4 is the operation of automated flexibility aggregation for optimal use of available distributed flexibility and maximization of profits for all participants in the portfolio (i.e. all Distributed FlexAssets and the aggregator entity itself). FlexContracts are agreed between end users and the aggregator, where users' preferences, constraints and compensation schemes are stated. Different approaches, leading to different mathematical models and algorithms are used for the optimal use of distributed flexibility assets (DFAs) and are shown via the development of different Use Case Scenarios (UCSs) as documented in previous D2.1 and D2.2 (in Month 4 and 6 respectively).

HLUC #4 focuses on the interaction between flexibility aggregators [1] and end energy prosumers[2]. Flexibility aggregators are considered as actors, which combine flexibility from energy prosumers and/or consumers and participate in markets as flexibility providers. The aggregated flexibility is sold to different stakeholders like DSOs, TSOs and BRPs, which participate in the flexibility markets as flexibility buyers (i.e. demand side of the flexibility market).

In previous D3.1[3], three research problems (one per UCS of HLUC #4) have been clearly defined. A high-level description of the three problems has taken place together with related works from the 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 have been presented.

This deliverable elaborates on the results of D3.1 by presenting the final version of mathematical modeling and proposed algorithms, while initial performance evaluation results are presented, too. Our next goal for M19-M26 period is to perform more simulations considering more realistic case studies and using real-life datasets by following the FLEXGRID data management plan.

[1] By the term "flexibility aggregator", we mean the market actor who aggregates distributed flexibility from numerous small-scale end energy prosumers. The main difference with the Energy Service Provider (ESP) actor (cf. FLEXGRID D4.1 w.r.t. to WP4 research work) that we use in FLEXGRID is that the ESP is a company that also owns several types of FlexAssets and thus does not only have a portfolio of end energy prosumers like the aggregator.

[2] With the term "end energy prosumer", we mean the end user who participates in a B2C flexibility market or has agreed on a FlexContract acting thus a customer of a flexibility aggregator company.

[3] https://flexgrid-project.eu/assets/deliverables/FLEXGRID_D3.1_final_version_29092020.pdf

1.2 Summary of state-of-the-art solutions for the aggregator's business challenges

In the previous D3.1, we have made an extensive survey work on the following state-of-the-art solutions, which are related with the flexibility aggregator's business challenges today as follows:

1) Survey on type of incentives for flexibility provisioning
2) Survey on novel market mechanisms for Demand Side Management (DSM) applications
3) Survey on intelligent S/W agent solutions and equipment for end energy prosumers
4) Survey on existing local flexibility markets in the EU area, where aggregators can trade their flexibility via a S/W platform

Regarding the first point above, there are several types of *incentive-based* and *price-based* energy or flexibility programs/contracts that exist in the real aggregators' business. More specifically, there are classical programs, in which consumers receive a fixed participation payment. There are also market-based programs, in which participants are rewarded based on their performance (e.g. amount of reduced electricity during critical conditions). The incentive-based flexibility contracts can also be categorized as: i) Direct Load Control, ii) Interruptible/Curtailable Load, iii) Emergency DSM Programs, iv) Capacity Market Programs, etc. On the other hand, there are considerably less price-based programs in the aggregator's business today. This is mainly due to the fact that advanced ICT infrastructure needs to be installed in each end prosumer's premises, which is a quite risky business plan. However, the idea of price-based flexibility programs/contracts has received much research attention from the international academic community, while many of these scientific solutions are being pilot tested in EU area during the last years.

Following up the idea of modeling price-based energy/flexibility contracts, the challenge is to model the objectives of both intelligent, foresighted end users and the aggregator in the wholesale market. In other words, it remains an open challenge to integrate intelligent algorithms for distributed FlexAssets' (DFAs) decisions with dynamic market mechanisms that are designed to serve the aggregators and ultimately the underlying physical electric grid. The afore-mentioned novel market mechanisms imply the existence of an ad-hoc B2C flexibility market, in which the aggregator has a dynamic interaction with all its end energy prosumers. The end energy prosumers are assumed to compete with each other in this new market in order for the aggregator to procure the cheapest possible flexibility, which will then be used to serve the power system's flexibility needs.

Finally, we have surveyed all existing real-life and conceptual local flexibility markets, which are being pilot-tested in the EU area during the last years. We have also compared them in terms of: i) the remuneration mechanism that they adopt for the flexibility aggregators (i.e. dispatch only or availability only or dispatch and availability payments), ii) the pricing rule that these markets adopt (i.e. pay-as-bid or pay-as-clear), iii) the flexibility products that are traded (i.e standardized vs. non-standardized flexibility products), and iv) the stakeholder that operates the local flexibility market (e.g. independent local market operator entity, DSO, platform co-designed by TSO and DSO, independent aggregator entity). For more details about the above-mentioned survey work, interested readers can refer to previous D3.1 (i.e. chapter 2 as well as section 3.2, 4.2 and 5.2 of D3.1).

1.3 Summary of research problems and FLEXGRID's research innovation

Following up the survey work mentioned above from both academic and industrial perspectives, we have come up with three main related FLEXGRID research problems, namely:

1) The aggregator wants to efficiently respond to a (set of) given FlexRequests made by a FlexBuyer (TSO/DSO, BRP) by optimally deciding the dispatch per FlexAsset/end energy prosumer (cf. UCS 4.1)
2) 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)
3) The aggregator wants to operate an ad-hoc B2C flexibility market with its end energy prosumers by employing advanced pricing models and auction-based mechanisms (cf. UCS 4.2)

Each one of the three research problems are described in detail in chapters 2-4 below. For each one of the three research problems, we present:

- Problem statement, related state-of-the-art and summary of FLEXGRID research contributions
- Proposed system model under study
- Problem formulation including the entire mathematical modeling
- Proposed algorithmic solution
- Simulation setup and performance evaluation results at TRL 3, which demonstrate and prove the concept of FLEXGRID's research innovations.
- Next steps on how to elaborate on the ongoing WP3 research work until M26 in order to test and validate the proposed mathematical models and algorithms with more realistic case studies and the use of real-life datasets.

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

In WP3, we focus on the scientific excellence of the proposed FLEXGRID services at TRL 3. The next goal is to adapt the most important WP3 scientific results 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).

More specifically, AFAT's frontend (GUI)[4] will be comprised of three basic tabs, namely:

- Manage a FlexRequest
- Create a FlexOffer
- Manage a B2C flexibility market

[4] AFAT's frontend services (GUI) will be developed by ETRA within WP6 context.

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

- A flexibility aggregation algorithm that tries to find the optimal scheduling solution of available FlexAssets in order to respond to a FlexRequest. The proposed solution is based on a centralized optimization model and it is described in detail in chapter 2 (cf. UCS 4.1).
- A flexibility aggregation algorithm that automatically aggregates availability flexibility in price/quantity tuples for a given future timeframe. The proposed solution is extensively described in chapter 3 (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 (cf. UCS 4.2).

Table 2 below clarifies 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.
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 balancing 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.

Within FLEXGRID project's context, we follow the NODES market paradigm and platform setup. Regarding technical and S/W development issues, we rely on 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. We assume an online flexibility marketplace (i.e. FLEXGRID ATP), in which the aggregator acts as a flexibility provider (i.e. FlexSupply side). We also consider that the aggregator is an independent market entity and has a portfolio of end energy prosumers. Each end energy prosumer has agreed a FlexContract with the aggregator that defines the terms under which the flexibility will be procured and remunerated. The aggregator registers all distributed FlexAssets in the marketplace, so that all other market stakeholders can see and verify them. The aggregator can use a set of intelligent mathematical models and algorithms to automate and dynamically adapt the flexibility aggregation process. This is exactly where FLEXGRID intelligence comes into the foreplay. The aggregator user will use the frontend and backend services of FLEXGRID's Automated Flexibility Aggregation Toolkit (AFAT). In the AFAT frontend, the aggregator user will be able to configure several input parameters and exhaustively run simulation scenarios in an online and offline mode as well as visualize the results via a user-friendly GUI. In the AFAT backend, all FLEXGRID WP3 algorithms will run. Part of this FLEXGRID intelligence (at TRL 5) will be open source, so that today and future aggregator's business can easily reuse it and potentially extend it.

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

Flexibility needs in the future are expected to increase and the benefits of developing local flexibility markets are the focus of the ongoing research. The potential of DER flexibility and their interaction in the current market design is limited, which is expected to change with the emerging role of the independent aggregator. A more detailed and extensive survey work on DERs, their flexibility and representation in the market can be found in chapter 2 and section 3.2 of previous FLEXGRID D3.1[5].

In this research problem, the aggregator needs to efficiently respond to a FlexRequest. This requires dispatch of FlexAssets within its portfolio of flexible DERs to activate the requested amount of energy. Response to a FlexRequest and dispatch of FlexAssets is decided in an online fashion and the aggregator needs the appropriate digital tools (supported by advanced mathematical models and algorithms) in order to minimize the risks and maximize its profits. The complexity of this problem stems from the uncertainty of future flexibility activation needs, the different operation patterns and behaviors of FlexAssets and the multiple types of FlexContracts.

In FLEXGRID, apart from maximizing the aggregator's profit, three main requirements are considered when responding and managing a FlexRequest that is generated by the market:

- **Ensure profit for all end-users.** The aggregator needs to have a strong and versatile portfolio to be able to fulfill FlexRequests. As the aggregator's portfolio consists of DERs of end-users, who agree to provide their flexibility, the aggregator user needs to ensure that all participating end users are properly incentivized and have tangible and well-quantified profits.
- **The aggregator respects reservations of flexibility.** Flexibility is needed for the system's secure and reliable operation. All reservations of the aggregator's flexibility need to be guaranteed and available when needed.
- **Deviations of scheduled operation**. Flexibility is defined as the ability to modify the operation pattern upon a request. This causes deviations to the already scheduled

[5] https://flexgrid-project.eu/assets/deliverables/FLEXGRID_D3.1_final_version_29092020.pdf

operation pattern (e.g. day-ahead energy market dispatch schedule). These deviations should occur only for Market Time Units (MTUs), where flexibility was requested from the market. Any other deviation would have undesired effects to BRPs and suppliers of the same FlexAssets.

2.2 System model

Flexibility as an individual product is currently not traded in any of the European electricity markets. Flexibility, as a property of an electrical supply or demand source, in current electricity markets, can be used by existing market players to better position themselves in the Day-Ahead market and qualify their assets for provision of ancillary services. Flexible assets are of a particular interest and in the current market design can participate in the part of the ancillary services market relating to balancing needs (frequency regulation) [6], the regulation market.

The regulation market in the existing regulatory framework can be divided in two stages: the DA reserve market and the balancing market (near real-time). In the DA reserve market, participants with successful/accepted bids commit to their ability to deliver the agreed quantity of energy for a specific Market Time Unit (MTU) and are in principle compensated for the reserved capacity (or else availability). In a second stage, at near real-time, the actual needs of the system become known and participants can bid for providing balancing energy. In most European markets, accepted bids of reserved capacity are obliged to offer balancing energy in the near-time balancing market. This however does not exclude the participation of other market players.

In the current electricity market design, the regulation market is under the supervision of the TSO, who is the sole buyer of reserve capacity and balancing energy at the transmission network level. The independent aggregator aims to represent flexibility of DERs in the market, which in turn will expand the potential of "flexibility" markets by attracting more flexibility buyers. Flexibility from DERs can be used to provide technical needs concerning system operation for both TSOs and DSOs, better balancing opportunities for trading (cf. BRP services) and requirements for non-dispatchable generation (cf. RES Producers) to participate in energy markets.

In the UCS examined in this chapter, the two stages of the regulation market are incorporated under the umbrella term of "FlexRequests". The independent aggregator is considered to be a price-taker market player, who interacts with the market by receiving FlexRequests-Reserve in the DA reserve market, and FlexRequests-Dispatch in the near real-time balancing market. Potentially, both types of FlexRequests can originate from multiple flexibility buyers, namely TSO, DSO and BRPs. The sequence of FlexRequests is shown in the figure below.

[6] ENTSOe, "Survey on Ancillary Services Procurement and Electricity Balancing Market Design", 2019, [Online]. https://www.entsoe.eu/publications/market-reports/#2019-surveys-on-ancillary-services-and-cross-border-balancing-initiatives

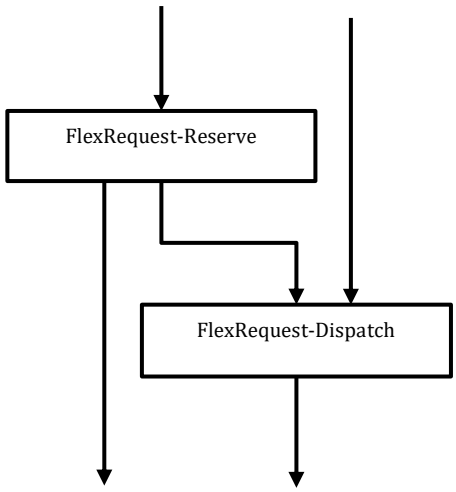

Figure 1: Diagram of potential sequence of types of FlexRequests

In case the aggregator accepts a FlexRequest of the reserve type (i.e. FlexRequest-Reserve), the reserved capacity needs to be available for the specific flexibility buyer for the required MTU. The aggregator is obliged to accept a FlexRequest-Dispatch from the same flexibility buyer for the specified MTU, up to the amount of capacity/energy of the FlexRequest-Reserve. In case of a FlexRequest-Dispatch with no prior reservation, the aggregator can either give a positive or negative response.

The scheduled operation (consumption/generation) of FlexAssets within the aggregator's portfolio, in accordance with the timeline and sequence of current trading floors for energy, is determined in the Day Ahead energy market (Supplier/Independent Aggregator). In other words, this means that we assume an already defined dispatch schedule generated after the market clearing process of the day-ahead energy market that should be (by all means) respected by all market participants.

A positive response to a FlexRequest-Dispatch requires the aggregator to activate a subset of FlexAssets of its portfolio to reach the requested amount of energy for the MTU. Depending on the cost function and with respect to the constraints of the FlexAssets and end-users, the aggregator issues for each FlexRequest-Dispatch the dispatch decision per FlexAsset for the relevant MTU.

As shown in the figure below, the clearing of the day-ahead reserve market precedes the near-real-time balancing market. The aggregator represents its portfolio of FlexAssets in both markets. Prior to any FlexRequest, which requires activation/dispatch of FlexAssets (Flex Request – Activation), the aggregator knows its obligations from the reserve market and from any other bilateral agreements (FlexRequest – Reserve).

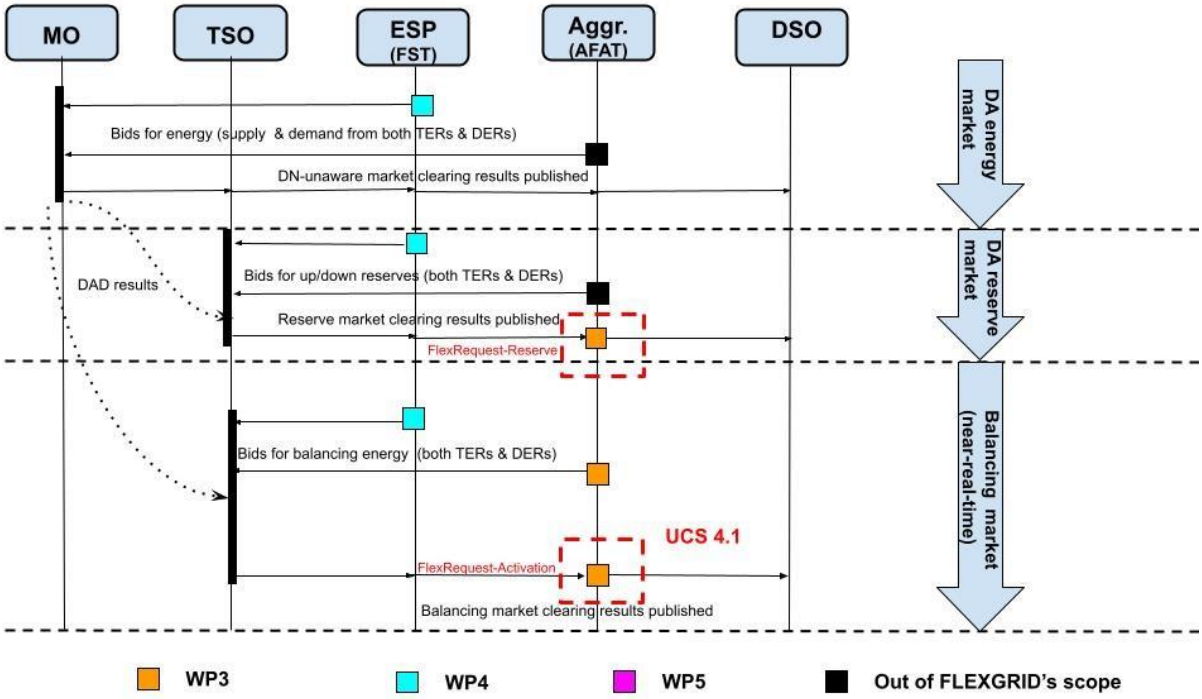

Figure 2: Placement of UCS 4.1 mathematical model and algorithm in the existing regulatory framework

2.3 Problem Formulation

The objective function, from the aggregator's point of view, is to maximize the profit from its participation in the flexibility market. This translates to the maximization of the revenues from the flexibility market (FlexibilityRevenues), while minimizing the associated costs (FlexibilityCosts).

The main parameters of the optimization problem required to calculate FlexibilityRevenues and FlexibilityCosts from the aggregator's perspective are:

- FlexRequests, which represent interaction with the market.
- FlexContracts, which represent interaction with end-users.
- FlexAssets that belong to the aggregator's portfolio.

2.3.1 FlexRequest

As stated in section 2.2, a FlexRequest can be one of two types: reserve or dispatch. The fields, parameters and information contained within FlexRequests are shown in the following table.

Table 3: Fields, parameters and information contained within FlexRequests

Type of Request	Reserve	Dispatch
FlexBuyer	X	X
Grid Location	A	A
Quantity type	Capacity (Power)	Energy
Quantity	Amount of kW	Amount of kWh
Regulation	Up/Down/Symmetrical	Up/Down

Reservation Price	Price per kW	-
Dispatch Price	Min price per kWh	Price per kWh
Acceptance Type	Full/Partial	Full/Partial
Timestamp	y ∈ [0,95]	y ∈ [0,95]
Market Time Unit (MTU)	x ∈ [0,95]	x ∈ [0,95]
Activation notice	Min time for dispatch request	-

Accepted FlexRequests of the reserve type, which represent interaction of the aggregator with the current DA reserve market, require availability of the requested capacity for the requested MTU. The aggregator must ensure that all control actions over its FlexAssets respect this constraint. The activation notice of the FlexRequest-Reserve provides also information of the latest timestamp, where the dispatch of reserved capacity can occur.

A FlexRequest of the reserve type offers reservation payments to the aggregator for the requested amount of available capacity/power. A FlexRequest-Reserve is assumed to contain information regarding the minimum price for activation per kWh, for a following FlexRequest-dispatch, which is necessary for the aggregator to decide to accept or not the reservation request.

The aggregator's revenue associated with reservation of FlexAssets for all MTUs within the time horizon T is:

$$\sum_{t \in T} \sum_{i \in FRr} C_{i,t}^{FRr} \cdot Pr_{i,t}^{FRr} \cdot x_{i,t}^{FRr} \qquad (2.1)$$

where FRr is the set of all FlexRequests-Reserve, $C_{i,t}^{FRr}$ and $Pr_{i,t}^{FRr}$ are respectively the requested capacity (kW) and price per kW of a FlexRequest-Reserve $i \in FRr$ for MTU t. Parameter $x_{i,t}^{FRr}$ denotes the acceptance type of the FlexRequest i and is equal to 1, if the FlexRequest is fully accepted.

A FlexRequest of the dispatch type can either follow a FlexRequest-Reserve, requesting activation of reserved assets, or come without a prior reservation. In the first case, the aggregator is required to fulfil the request, while in the second case, the response of the aggregator can be either positive or negative (cf. Figure 1 above). In any case, the aggregator's revenue associated with requests for dispatch of flexibility depends only on the activated flexible energy and does not include reservation payments:

$$\sum_{t \in T} \sum_{j \in FRd} E_{j,t}^{FRd} \cdot Pd_{j,t}^{FRd} \cdot x_{j,t}^{FRd} \qquad (2.2)$$

where FRd is the set of all FlexRequests-Dispatch, $E_{j,t}^{FRd}$ and $Pd_{j,t}^{FRd}$ are respectively the requested energy (kWh) and price per kWh of a FlexRequest-Dispatch $j \in FRd$ for MTU t. Parameter $x_{j,t}^{FRd}$ denotes the acceptance type of the FlexRequest j and is equal to 1 if the FlexRequest is fully accepted.

Reserved capacity and activated energy for a MTU are related through:

$$E_{i,t}^{FRd} = Cd_{i,t}^{FRr} \cdot \tau \qquad (2.3)$$

where τ is the duration of a single MTU.

2.3.2 FlexContract/End-user compensation

End-user compensation, or else the aggregator's cost of acquiring flexibility, is defined in the FlexContracts of the aggregator with the end-users that belong to its portfolio. The compensation of the end-user can be separated into three categories: Reservation/Participation, Dispatch and Activation.

The *reservation/participation* component provides the incentive to the end-user to participate in the aggregator's portfolio. This ensures a decrease of the end-user's electricity bill, even in the case where none of his flexibility assets are dispatched/used. The *participation payment* depends on the amount of flexibility that the end-user provides to the aggregator. In any case, concerning the time horizon of the optimization problem, this is a fixed cost for the aggregator.

The *dispatch component* is related to the dispatched flexibility/energy and it is energy-dependent. In contrast, the *activation component* is related to the number of times that a FlexAsset is activated and depends on the number of activations. For a given FlexAsset, the compensation involving its use can either be energy dependent or dependent on the number of activations. Energy dependent compensation for the FlexAssets of end-users N over the time horizon T is computed based on the following mathematical formula:

$$\sum_{t\epsilon T}\sum_{n\epsilon N}\sum_{k\epsilon FA} E_{k,t,n}^{FA} \cdot Pr_{k,t,n}^{FA} \qquad (2.4)$$

Here, $E_{k,t,n}^{FA}$ is the activated flexible energy of FlexAsset $k \epsilon FA$ and $Pr_{k,t,n}^{FA}$ is the price per kWh for MTU t.

The activation component of compensation, which depends on number of activations is:

$$\sum_{t\epsilon T}\sum_{n\epsilon N}\sum_{k\epsilon FA} Pr_{k,t,n}^{FA}(l) \cdot x_{k,t,n}^{FA} \qquad (2.5)$$

In this case, $Pr_{k,t,n}^{FA}(l)$ is the price for the l^{th} activation and $x_{k,t,n}^{FA}$ is 0 or 1 depending on the status of activation of FlexAsset $k\epsilon FA$ during MTU t.

Activation and dispatch payments of end-user compensations are one of the aggregator's costs, which depend on the dispatch decision of the aggregator. The objective of the aggregator is through the optimal management of its portfolio to minimize these payments, thus its cost. The participation fee ensures that end-users are properly incentivized to provide their flexibility. Activation and dispatch payments described in FlexContracts provide sufficient compensation for end-users and do not depend on the value of flexibility in the market. The risk of participating in flexibility market (cf. possible imbalances) is undertaken by the aggregator. The compensation of end-user $n \epsilon N$ for participation in the flexibility market is:

$$PF(n) + \sum_{t \epsilon T} \sum_{k \epsilon FA} E^{FA}_{k,t,n} \cdot Pr^{FA}_{k,t,n} + Pr^{FA}_{k,t,n}(l) \cdot x^{FA}_{k,t,n} \qquad (2.6)$$

A constraint of the independent aggregator's optimization problem is that all participating end-users should benefit from the provision of their FlexAssets. The baseline cost of electricity of an end user is:

$$\sum_{t \epsilon T} E_b(t) \cdot rp(t) \qquad (2.7)$$

where $E_b(t)$ and $rp(t)$ denote the energy of the baseline consumption and the retail price of electricity at MTU t. Participation in the flexibility market leads to deviations between scheduled/baseline operation (E_b) and actual operation (E_a). Thus, the aggregator's constraint concerning an end-user is:

$$\sum_{t \epsilon T} E_a(t) \cdot rp(t) - \left\{ PF(n) + \sum_{t \epsilon T} \sum_{k \epsilon FA} E^{FA}_{k,t,n} \cdot Pr^{FA}_{k,t,n} + Pr^{FA}_{k,t,n}(l) \cdot x^{FA}_{k,t,n} \right\} \leq \sum_{t \epsilon T} E_b(t) \cdot rp(t) \qquad (2.8)$$

and should stand for all end-users $n \, \epsilon \, N$.

2.3.3 FlexAssets

Apart from the end-user compensation, FlexContracts contain information regarding end-users' preferences and constraints, FlexAssets and their flexibility behavior. This information is necessary for the aggregator to extract the available flexibility and cost for each FlexAsset and construct tables for relevant MTUs.

Moreover, it is assumed that the scheduled operation of each individual FlexAsset is known and the appropriate ICT infrastructure is available to allow monitoring and direct control of all FlexAssets.

Flexibility assets can either be supply, demand or storage assets. All types of FlexAssets, depending on the scheduled operation pattern (cf. DA energy market), can be used in principle for either direction of a FlexRequest (i.e. up or down regulation).

A more interesting classification of assets is based on the type of control over their operation pattern. A fully flexible asset is labeled as "**adjustable**" and its basic property is the ability to activate the potential flexibility, defined by both technical characteristics and user constraints and preferences, based only on current operation for any given timeslot (MTU), without any dependency on past operation and without affecting operation at future time slots. This type of FlexAssets can be associated with either dispatch or activation payments.

Assets with shiftable operation patterns belong to another type, called "**shiftable**". The control of assets in this category allows shifting the operation pattern to a past or future timeslot with respect to the scheduled operation pattern. The total energy consumed/ generated by this type of assets is defined and the operation pattern always includes consecutive MTUs. When the operation pattern of assets can be shifted in time, but can also

be interrupted, the asset is "**shiftable-interruptible**". The total amount of energy is the same, but the operation can be divided along the time horizon under consideration.

The operation pattern of both shiftable and shiftable-interruptible FlexAssets is considered to involve more than one MTUs, thus shifting their operation affects multiple MTUs and causes deviations from the baseline schedule, which were not necessarily requested. These deviations may incur extra cost, depending on the market structure. In the case, where all deviations due to shifts can be submitted to the market and establish a new scheduled operation pattern/baseline, there are no extra costs. It is possible however, that for MTUs in the near future, there is no possibility for the aggregator to submit a new schedule and imbalance costs for those MTUs need to be considered.

The aggregator has two alternatives to deal with imbalance costs. The first one involves the management of the operation of FlexAssets within its portfolio to absorb undesired deviations and reach the baseline of the DA energy market for MTUs, where flexibility was not requested, and rescheduling is not an option. The second one is the acceptance of imbalance costs imposed by the market.

2.3.4 Objective Function of the Aggregator

The objective function of the aggregator is to maximize the profit from participating in the flexibility market by responding to multiple FlexRequests within a given time horizon T. The objective function to be maximized, including only variable components, can be formulated as follows:

$$\sum_{t\in T}\sum_{j\in FRd} E_{j,t}^{FRd}\cdot Pd_{j,t}^{FRd}\cdot x_{j,t}^{FRd} - \sum_{t\in T}\sum_{k\in FA} c_{k,t}^{FA}\left(e_{k,t}\right) - \sum_{t\in T} IC\left(SO(t)+E(t)-AO(t)\right) \qquad (2.9)$$

where the first term represents revenues from FlexRequests-Dispatch, the second one the cost for activating flexibility within the portfolio and the third one imbalance costs imposed by the market. Technical characteristics, user preferences and constraints, FlexContracts and the baseline consumption of FlexAssets determine the cost $c_{k,t}^{FA}\left(e_{k,t}\right)$ of acquiring flexible energy e from FlexAsset $k \in FA$ at MTU t. The imbalance cost IC is the market price for undesired deviations of scheduled energy, which per MTU t are equal to scheduled operation ($SO(t)$) plus requested and accepted flexibility ($E(t)$) minus actual/measured operation ($AO(t)$).

2.4 Algorithmic solution

The aggregator's objective function is a multi-level optimization problem with constraints, where the aggregator needs to optimally schedule its assets for each MTU in order to maximize the profit over the entire time horizon. The output of this evaluation scenario is the set of activated FlexAssets for each MTU.

All the requests for dispatch of flexibility within the time horizon are not known in advance, thus at run-time, during online operation of the process "Manage a FlexRequest" that is available in the Automated Flexibility Aggregation Toolkit (AFAT), the aggregator has limited

information of future requests. This limited information prohibits exhaustive search over the entire time horizon, which is only possible for offline operation and "what-if" simulation scenarios. More details about the S/W integration of UCS 4.1 algorithm inside AFAT and the respective AFAT's frontend and backend services are provided in chapter 5 of this report.

Upon a request for dispatch of energy at t_0 for MTU t, the proposed approach is for the aggregator to select for dispatch the lower cost FlexAssets in order to reach the necessary amount of energy (greedy scheduling technique). Feasibility of a potential set of FlexAssets would require respect over the constraints (reserve availability). The cost assigned to each FlexAsset takes into account activation cost for the MTU under consideration, modification of scheduled operation and effects on availability in future time slots. The weight of the component involving availability in future timeslots depends on the probability of having a FlexRequest on each MTU. These probabilities are determined based on scenarios of FlexRequests. As the portfolio of the aggregator can be quite extended, classification of FlexAssets by using clustering techniques can be applied to improve performance and scalability.

2.5 Simulation setup and performance evaluation results

2.5.1 Simulation setup

The basic inputs of this UCS are FlexRequests, FlexContracts and FlexAssets. As flexibility markets do not exist in the current regulatory market design, FlexRequests of the reserve type will be based on the existing DA reserve market. Volumes and prices of FlexRequests of dispatch type will be based on data of existing balancing energy markets.

Operation patterns of FlexAssets will be taken from load monitoring of different DERs. Cost aspects involving activation of flexibility will be derived from real-life explicit DR programs/business cases and relevant research work. Imbalance costs of the market will be based both on prices of intraday markets and balancing markets.

The length of the time horizon of the optimization problem is a 24-hour day, (00:00-23:59), which coincides with the output of the existing DA energy market and input datasets will be used for several days to observe the efficiency of the algorithm and to determine the optimal parameters for cost and selection functions.

2.5.2 Performance evaluation and KPIs

The principal goal of this research problem is the maximization of the aggregator's profit for offering flexibility in the electricity market. In order to evaluate the performance of the proposed algorithm, the following KPIs will be measured:

- **Reliability of the aggregator towards flexibility reservations and activations.** The aggregator must be consistent and reliable towards its actions in the market. Incompetence to activate energy that has been requested will lead to penalties and extra costs.
- **Utilization of flexibility of portfolio.** The aggregator needs to utilize, either through reservations or activations, the flexibility of its portfolio in an optimal way. If large

amounts of flexibility remain unused for multiple time horizons, the cost of sustaining the portfolio is not justifiable and profitable.

- **Undesired deviations from scheduled operation.** The aggregator should prioritize absorbing deviations with actions within its portfolio. Imbalances in the market indicate the need to expand the portfolio of FlexAssets.

2.6 Next research steps for M19-M26 period

During the period M19-M26 of the FLEXGRID project, the focus of this research problem will be on creating realistic datasets and test performance of the proposed approach for several types of flexibility portfolios and cost functions.

A version of the UCS 4.1 algorithm will be integrated in the Automated Flexibility Aggregation Toolkit (AFAT) and FLEXGRID ATP, which will be done in close collaboration with the work performed in WP6. The aggregator user will be able to visualize its portfolio and the use of the proposed algorithm will allow the issue of an optimized dispatch of FlexAssets. Two types of operation will be possible, namely online and offline. During online operation, the aggregator will make decision concerning its FlexAssets when receiving a FlexRequest. For offline operation, the aggregator will be able to run different “what-if” simulation scenarios to test its ability to manage more FlexRequests by altering accepted reservation of flexibility and by expanding its portfolio in the future.

3 An aggregator maximizes its revenues by dynamically orchestrating distributed FlexAssets from its end users to optimally participate in near-real-time energy markets

This chapter deals with the research problem of FLEXGRID's UCS 4.3. In this UCS, we consider the problem of an aggregator that wants to offer aggregated flexibility in a near-real-time energy market on behalf of a vast number of distributed and small-scale FlexAssets. In today's EU electricity markets, this near-real-time market is the so called "balancing energy" market, which is operated by the TSO. However, within FLEXGRID project's scope, we also consider near-real-time distribution level flexibility market (DLFM) in which balancing energy product is traded between the DSO and aggregators that offer distributed local flexibility to the DSO. Thus, the DSO is able to deal with local congestions and imbalances that may come up in near-real-time contexts.

Following up the research work of UCS 4.1 that was presented in the previous chapter, we now focus on another aggregator's challenging task, which is the design of the aggregator's FlexOffer strategy in a near-real-time context. More specifically, it is difficult for the aggregator to capture the flexibility cost of a portfolio of FlexAssets within a price-quantity offer, since the costs and constraints of FlexAssets exhibit inter-temporal dependencies.

In FLEXGRID UCS 4.3 research work, we propose **a generic method for constructing aggregated FlexOffers that best represent the aggregator portfolio's actual flexibility costs, while accounting for uncertainty in future timeslots**. For the case study presented, we use offline simulations to train and compare different machine learning algorithms that receive the information about the state of the FlexAssets and calculate the aggregator's FlexOffer. Once trained, the machine learning algorithms can make fast decisions about the portfolio's FlexOffer in the near-real-time balancing market. The performance evaluation results show that the proposed method performs reliably towards minimizing the aggregator's imbalances.

In FLEXGRID ATP, the aggregator user will be able to utilize the Automated Flexibility Aggregation Toolkit (AFAT) to make efficient FlexOffer in near-real-time balancing markets and DLFMs. In the online operation mode, the aggregator can automatically create 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). In the offline operation mode, 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 balancing market (i.e. DLFM). If the FlexOffer is not accepted in DLFM, it can be automatically forwarded to the TSO's balancing market.

3.1 Problem statement, related state-of-the-art and FLEXGRID research contributions

Within the previous deliverable D3.1[7], we have conducted an extensive survey work on the related state-of-the-art research that has taken place during the last years in this research field. The interested reader may search for more details in the respective section of D3.1 and the references therein.

Summarizing this international literature review, studies typically assume some type of electricity price forecast, and a price-taking aggregator entity that only bids an energy quantity (much like a supplier does) instead of price-quantity pairs. Creating price-quantity pairs, is a challenging task for the aggregator, since the costs and constraints of its DERs have inter-temporal couplings, i.e., the flexibility cost of a DER in the current timeslot is dependent on how the DER flexibility will be controlled in future timeslots. Moreover, the aggregator's bid must be decided in an online (i.e. near-real-time) fashion, which means that the available time for computations is very limited (e.g. 15 minutes, 5 minutes or even less in the future). The problem gets even more complex, if we want to take into consideration all the various and diversified FlexAsset models like electric vehicles, heat pumps, different types of storage units, shiftable loads, etc.

In FLEXGRID, we take into consideration four main requirements towards designing an aggregator's FlexOffer as follows (cf. also 16):

- **Req #1: The aggregator's FlexOffer should be concise.** Given the scale of aggregators and the complexity of the constraints of FlexAssets, it is impossible to communicate precise information about every FlexAsset. Instead, aggregate flexibility feedback must be a concise summary of a system's constraints. Even if it was possible, providing exact information about the constraints of each FlexAsset governed by the aggregator would not be desirable because the FlexAsset constraints are typically private. Information conveyed to the system operator must limit the leakage about specific FlexAsset constraints[8].
- **Req #2: The aggregator's FlexOffer should be informative.** The feedback sent by an aggregator needs to be informative enough that it allows the system operator to achieve operational objectives, e.g., minimize cost, and, most importantly, guarantee the feasibility of the whole system with respect to the private FlexAsset constraints.
- **Req#3: The aggregator's FlexOffer should be general enough.** Any design for an aggregator's FlexOffer must be general enough to be applicable for a wide variety of controllable loads, e.g., electric vehicles (EVs), heating, ventilation, and air conditioning (HVAC) systems, energy storage units, thermostatically controlled loads, residential loads, heat pumps, etc. It is impractical to imagine a different FlexOffer for each FlexAsset, so the same design must work for all types of distributed FlexAssets.

[7] https://flexgrid-project.eu/assets/deliverables/FLEXGRID_D3.1_final_version_29092020.pdf

[8] In this document, we focus our mathematical modeling on the TSO's balancing market. A similar approach may be followed for an aggregator's FlexOffer in the novel distribution-level flexibility markets (DLFMs) proposed by FLEXGRID. In this case, the aggregator may provide near-real-time balancing energy services to the local DSO.

- **Req #4: The aggregator's FlexOffer should be real-time.** The system is time-varying and non-stationary. So it is crucial that (nearly) real-time feedback can be defined and approximated if it is to be used in online FlexOffers made by the aggregator.

Summarizing FLEXGRID's scientific contributions, we propose a generic method for capturing the aggregator's upward and downward flexibility cost for a set of distributed FlexAssets (or else DERs) that have non-convex models and inter-temporal couplings. The method is model-free in the sense that it is not tailored to any specific DER model. Rather, it can be applied to any type of FlexAssets, regardless of the specific model that each type of FlexAsset has. We use a fitting function for this purpose. In order to address the uncertainties of the FlexAssets' parameters, we perform offline scenario-based simulations, and use these simulations to train a machine-learning (ML) algorithm. Different ML methods are tested and compared. In online operation, the trained ML can be provided with the current state of the FlexAssets, and predict the optimal aggregator's FlexOffer (prices for given levels of balancing energy) for the next timeslot ahead very quickly and, as our simulation results indicate (cf. section 3.5 below), with very good accuracy.

3.2 System model

There is a general consensus that participation of distributed FlexAssets (also called "DERs") should be realized via aggregators, i.e., entities that participate in electricity markets and undertake balance responsibility on behalf of a portfolio of multiple DERs/FlexAssets. A FlexAsset/DER is assumed to be registered with an aggregator[9], where the latter installs the necessary communication infrastructure that allows it to monitor, forecast and control the electricity profile of the FlexAsset. Each FlexAsset has a certain set of preferences towards its electricity profile, as well as a cost function that maps a FlexAsset's electricity profile to a monetary cost. For example, an Electric Vehicle (EV) has an arrival time and a certain energy that it needs to receive (charge) before its departure. If the aggregator requests the EV to receive less energy than required, then the EV requests a compensation for this flexibility service.

Market participants (buyers and sellers) can trade energy in the day-ahead and/or intra-day markets. This free trade stops at a certain time before real time (delivery time) in order for the system operator to ensure that the system will be balanced in real time operation. The time in which trading stops is called gate closure time. After the gate closure, each participant reports a certain energy profile (energy bought/sold) to the system operator. This profile is referred to as the participant's market program.

In real-time operation, the transmission system operator (TSO) is responsible for maintaining the balance between supply and demand. Given a market program for each market participant (cf. day-ahead dispatch – DAD results in the figure below), the TSO receives the players' offers for providing or requesting balancing energy. A cost optimization problem is run at the TSO's side, through which the balancing energy dispatch of each player is determined.

[9] In FLEXGRID ATP, there is an API via which the aggregator can register all its contracted FlexAssets/DERs in the platform, so that the latter can participate in the various B2B and B2C electricity markets.

In order for the TSO to be able to solve this optimization problem in a fast and scalable way, the balancing energy offers (i.e. FlexOffers) made by the participants need to be provided in a certain bidding format, which guarantees that the optimization problem is tractable. This FlexOffer process is depicted in the red outlined area in the figure below. For example, a participant is typically required to make an offer for upward balancing energy and downward balancing energy for the timeslot ahead. A typical FlexOffer is a mapping that relates a level of balancing energy provision to a certain monetary cost. These FlexOffers are typically required to be in a step-wise form, i.e., pairs of price-quantity.

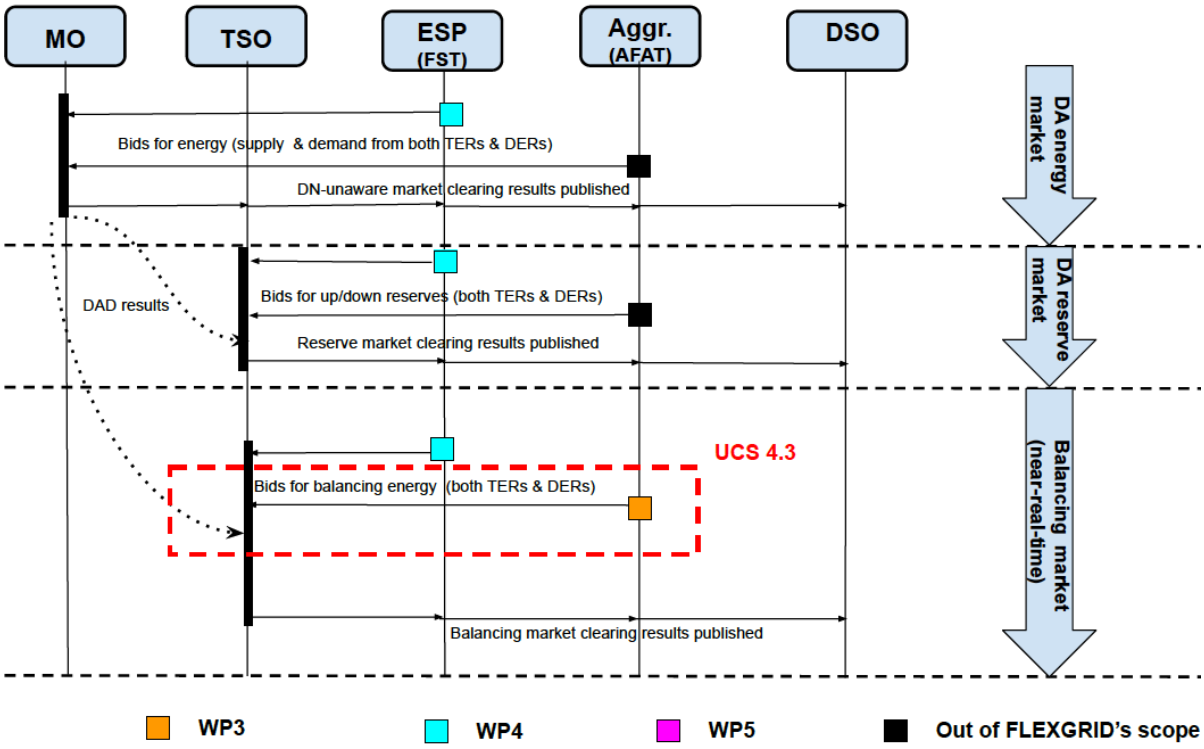

Figure 3: Placement of UCS 4.3 mathematical model and algorithm in the existing regulatory framework[10]

We consider an aggregator entity that is responsible for submitting offers for balancing energy on behalf of its portfolio. Let N denote the set of FlexAssets registered with the aggregator and T denote a set of timeslots within a particular time horizon. The electricity demand of N in T is comprised by a vector $\mathrm{P}_N = \{\mathrm{P}_N^1, \mathrm{P}_N^2, \ldots \mathrm{P}_N^{|T|}\}$, where an element P_N^t represents the portfolio's market program (i.e. the energy bought in the day-ahead energy market represented by the term "DAD results" in the figure above) for timeslot t. In other words, we assume that the aggregator should respect a day-ahead energy schedule, which is the result of the day-ahead energy market clearing process. We assume that this day-ahead energy schedule should be respected by the aggregator.

Some FlexAssets can offer certain flexibility with respect to their electricity demand. In particular, the electricity consumption of a flexible DER n in timeslot t can be controlled. We

[10] A similar approach for the creation of the aggregator's FlexOffer is followed for the proposed near-real-time DLFMs that are proposed within FLEXGRID. More details about the respective sequence diagrams and regulatory assumptions are provided in chapter 2 of D6.1 (M18) - https://flexgrid-project.eu/deliverables.html

denote this control variable as x_n^t and the respective vector $\mathbf{x}_n = \{x_n^1, x_n^2, \ldots, x_n^{|T|}\}$ denotes the controllable consumption profile of a flexible DER across the time horizon. A certain consumption profile $\mathbf{x}_n$, generally comes with a cost for FlexAsset n. More specifically, a FlexAsset's cost is defined as a function $c_n(\mathbf{x}_n)$. Moreover, FlexAsset n bears a set of constraints regarding its profile, which for the moment are abstractly denoted as:

$$\mathbf{x}_n \in F_n. \qquad (3.1)$$

Here, a major observation that have to be carefully considered is that constraints (3.1) may couple a variable $x_n^{t_1}$ with a variable $x_n^{t_2}$, i.e., a FlexAsset's model may exhibit inter-temporal couplings. The aggregated flexible consumption in timeslot t is denoted as X_N^t, where

$$X_N^t = \sum_{n \in N} x_n^t \qquad (3.2)$$

and the respective vector $\mathbf{X}_N$ denotes the aggregator's flexible consumption profile across the time horizon. The difference $\mathrm{P}_N^t - X_N^t$ is the aggregator's provided balancing energy in timeslot t. Note that it can also take on negative values when the aggregator "absorbs" more energy than P_N^t.

The aggregator has to provide a FlexOffer in the timeslot τ ahead (i.e. a cost for energy injection and an offer for energy absorption). The bidding format is subject to the rules of the TSO. Typically, it has to be in a form of a step-wise function that defines pairs of balancing energy quantity and price as it is shown in the figure below in order to make the economic dispatch problem solvable by standard mixed-integer programming techniques. The above bidding format, although conducive for the TSO, is quite restrictive for the aggregator, since it cannot fully capture the aggregator's actual model, which is comprised by the cost functions $c_n(\mathbf{x}_n)$ and constraints $\mathbf{x}_n \in F_n$ of all FlexAssets in the aggregator's portfolio. Note also, that the FlexAssets' cost functions and constraints may exhibit inter-temporal dependencies, too.

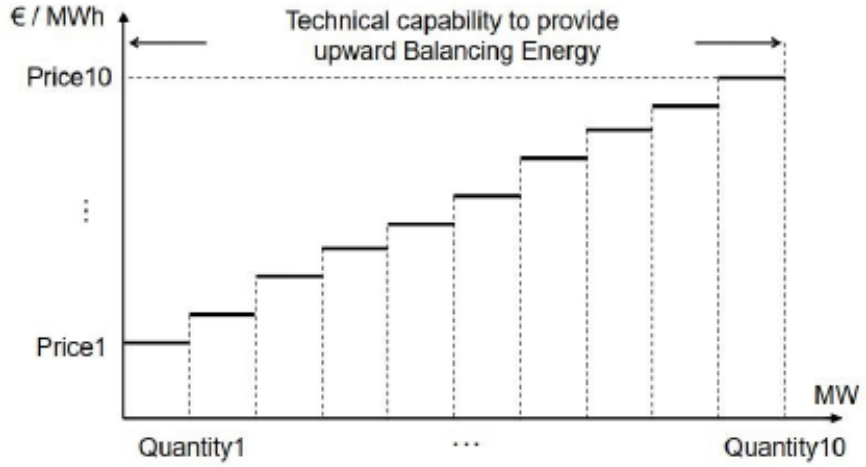

Figure 4: A typical form of an aggregator's FlexOffer for the upward balancing energy product[11]

[11] A similar FlexOffer curve is also created by the aggregator for the downward balancing energy product.

3.3 Problem Formulation

In order to facilitate the method presentation, and without loss of generality with respect to the methods that will be presented, we assume that the aggregator offers one price-quantity pair, i.e., a per-unit price b_{up}^{τ} paired with a maximum quantity $\mathrm{B}_{\mathrm{up}}^{\tau}$ for upward balancing energy (i.e. injecting power by curtailing electricity consumption) and bids a per-unit price b_{down}^{τ} and a maximum quantity $\mathrm{B}_{\mathrm{down}}^{\tau}$ for downward balancing energy in the next timeslot τ (i.e. buying more energy than P_N^{τ}). The method can be directly extended to as many pairs as desirable, since our proposed method is generic as it will be clarified shortly.

The mathematical form of the Aggregator's FlexOffer reads as:

$$q^{\tau}(X_N^{\tau}) = \begin{cases} b_{up}^{\tau}(\mathrm{P}_N^{\tau} - X_N^{\tau}) & , \quad \mathrm{P}_N^{\tau} - X_N^{\tau} \geq 0 \quad (3.3a) \\ b_{down}^{\tau}(X_N^{\tau} - \mathrm{P}_N^{\tau}) & , \quad \mathrm{P}_N^{\tau} - X_N^{\tau} < 0 \end{cases}$$

$$\mathrm{P}_N^{\tau} - X_N^{\tau} \leq \mathrm{B}_{\mathrm{up}}^{\tau}, \quad \mathrm{P}_N^{\tau} - X_N^{\tau} \geq 0 \qquad (3.3b)$$

$$X_N^{\tau} - \mathrm{P}_N^{\tau} \leq \mathrm{B}_{\mathrm{down}}^{\tau}, \quad \mathrm{P}_N^{\tau} - X_N^{\tau} < 0. \qquad (3.3c)$$

where $q^{\tau}(X_N^{\tau})$ is the Aggregator's cost function (for upward and downward balancing energy) and (3.3b) and (3.3c) communicate to the TSO that the aggregator can receive a dispatch up to $\mathrm{B}_{\mathrm{up}}^{\tau}$ ($\mathrm{B}_{\mathrm{down}}^{\tau}$) for balancing energy up (down).

The TSO gathers all the bids for balancing energy, including the bid (3.3a)-(3.3c) of the aggregator and the set of bids $\mathcal{B}^{\tau}$ of other market participants, and clears the balancing market close to real-time[12] by solving an economic dispatch problem that minimizes the system's cost SC. The output of the economic dispatch problem is the balancing energy dispatch decisions for each market participant and the balancing energy prices $\lambda_{\mathrm{up}}^{\tau}$, $\lambda_{\mathrm{down}}^{\tau}$ for upward and downward balancing energy respectively. Note that the system either dispatches upward or downward balancing energy, so only one of the two prices is non-zero. We denote the balancing energy dispatch of the aggregator as D^{τ}. Economic dispatch problem of the TSO reads as:

$$\min \{SC\} \qquad (3.4)$$
$$s.t.\ (3.3a) - (3.3c), \mathcal{B}^{\tau}$$

Thus, the aggregator's dispatch order D^{τ} depends on its FlexOffer $b_{up}^{\tau}, b_{down}^{\tau}$ through problem (3.4).

Upon receiving the dispatch order and the price, the aggregator calculates the power of each FlexAsset so as to maximize its profit π. The procedure is illustrated in the figure below. The aggregator receives a revenue $\lambda_{\mathrm{up}}^{\tau} \cdot \max\{0, (\mathrm{P}_N^{\tau} - X_N^{\tau})\}$ from providing balancing energy up (or a cost $\lambda_{\mathrm{down}}^{\tau} \cdot \max\{0, (X_N^{\tau} - \mathrm{P}_N^{\tau})\}$ for down), while in case the aggregator deviates from the TSO's dispatch order, it receives a penalty $\lambda_{\mathrm{Imb}} \cdot |X_N^{\tau} - D^{\tau}|$. Finally, the aggregator pays a cost $\sum_{n \in N} c_n^{\tau}(x_n^{\tau})$ to its FlexAssets in order to shape their profile to $\{\mathrm{x}_n\}_{n \in N}$ such that

[12] In EU's nordic countries, this timeframe is usually 1 hour, but it is expected to be 15 minutes within the next couple of years. The ambition is to be as closest to real time as possible in the future, so the proposed mathematical model and machine-learning based algorithmic solution is expected to have an even greater impact in the future.

equation (3.2) holds. Based on the above, the aggregator's profit in current timeslot τ, can be expressed as:

$$\pi^{\tau} = \lambda^{\tau}_{\text{up}} \cdot \max\{0, (\mathrm{P}^{\tau}_{N} - X^{\tau}_{N})\} - \lambda^{\tau}_{\text{down}} \cdot \max\{0, (X^{\tau}_{N} - \mathrm{P}^{\tau}_{N})\} - \lambda_{\text{Imb}} \cdot |X^{\tau}_{N} - D^{\tau}| - \sum_{n \in N} c^{\tau}_{n}(x^{\tau}_{n}) \quad (3.5)$$

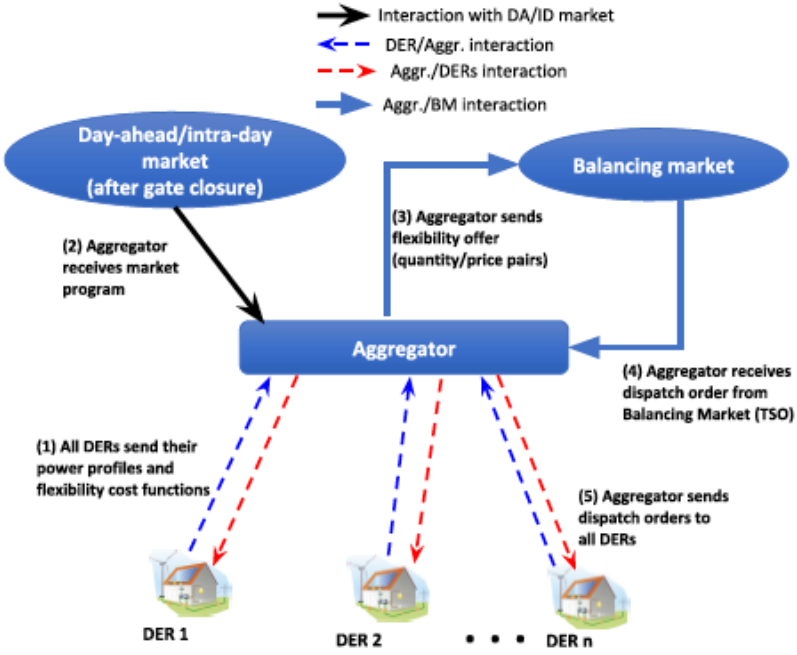

Figure 5: Main steps for the realization of FLEXGRID UCS 4.3 in the existing regulatory framework

The aggregator deals with a sequential decision-making problem where, in the first stage of current timeslot τ, it decides upon its FlexOffer $b^{\tau}_{\{up,down\}}$, and in the second stage (after receiving its dispatch order), it decides upon the electricity consumption of its DERs $\{x^{\tau}_{n}\}_{n \in N}$ and consequently X^{τ}_{N}. The decisions are realized and the procedure repeats in the next timeslot. In the first stage decision, the aggregator's objective is to find the optimal FlexOffer $b^{\tau}_{\{up,down\}}$ that maximizes its expected profit over the second stage decision $\{x^{\tau}_{n}\}_{n \in N}, X^{\tau}_{N}$ and also over the expected profits of future timeslots. This is formalized through a multi-stage stochastic optimization problem that takes a nested form:

$$\max_{b^{\tau}_{up}, b^{\tau}_{down}} \left\{ \mathbb{E}\left[\max_{\{x^{\tau}_{n}\}_{n \in N}, X^{\tau}_{N}} \pi^{\tau} \right] + \mathbb{E}\left[\max_{b^{\tau+1}_{up}, b^{\tau+1}_{down}} \left\{ \mathbb{E}\left[\max_{\{x^{\tau+1}_{n}\}_{n \in N}, X^{\tau+1}_{N}} \pi^{\tau+1} \right] + \ldots \right\} \right] \right\}$$
$$\text{s.t. } (1) - (5) \quad (6)$$

where the expectations are over dispatch orders D^{t} and prices λ^{t}_{up}, $\lambda^{t}_{\text{down}}$, that depend on decisions $b^{\tau}_{up}, b^{\tau}_{down}$ through problem (3.4).

Since the aggregator has no information on the bids $\mathcal{B}^{t}$ of other players for the current or future timeslots, it cannot tackle problem (3.6) optimally, since it cannot have an expression

for the dispatch orders or prices. In what follows, we propose a method through which the aggregator can handle this uncertainty upon deciding its FlexOffers in the first stage.

Let us consider a set S of arbitrary scenarios $s \in S$ for the aggregator's dispatch orders, constrained by (3.3b) and (3.3c) over the entire horizon T. Let the sequence of dispatch orders for a certain scenario s be denoted as $\mathbf{D}_s = \{D_s^1, D_s^2, \ldots, D_s^{|T|}\}$. We consider a conservative strategy, where, given the dispatch information, the aggregator opts for minimizing its total balancing energy and imbalance costs, as in:

$$C_s^* = \min_{\boldsymbol{x}_n, \boldsymbol{X}_N} \left\{ \sum_{n \in N} c_n(\boldsymbol{x}_n) + \sum_{t \in T} \lambda_{\text{Imb}} \cdot |X_N^t - D_s^t| \right\} \quad (7)$$
$$\text{s.t. } (1), (2)$$

Using problem (3.7), we can obtain the optimal variables $\mathbf{X}_{N,s}^{t,*}$ and respective optimal costs C_s^* for each scenario s. Then, for each scenario, we fix the values of $\mathbf{X}_{N,s}^{t,*}$ and C_s^*, and solve a fitting problem to decide the variables b_{up}^t, b_{down}^t for the entire horizon, such that the distance between the average aggregator's cost given by problem (3.7) and the cost given by the aggregator's FlexOffer function (3.3a), is minimized:

$$\min_{b^t_{\{up,down\}}} \left\{ \sum_{s \in S} \left(\sum_{t \in T} q^t(X_{N,s}^{t,*}) - C_s^* \right)^2 \right\} \quad (8)$$
$$\text{s.t. (3a).}$$

Using the above method, the Aggregator can retrieve a mapping from input data $\mathrm{P}_N, c_n(\mathbf{x}_n), F_n$ to the decision for its FlexOffers $b^{\tau}_{\{up,down\}}$. The FlexOffer estimation method is summarized in Algorithm 1 below. However, a large number of scenarios may be required before a good approximation is achieved, which can be impractical for real-time operation. Thus, a Machine Learning (ML) based solution to this problem is described in the next section.

Algorithm 1 Bid estimation method

1: Read input $\mathbf{P}_N, c_n(\boldsymbol{x}_n), F_n$
2: Create a set S of scenarios for dispatch sequences $\boldsymbol{D}_s$
3: **for** $s \in S$
4: solve problem (7)
5: store values $\boldsymbol{X}_{N,s}^{t,*}$, C_s^*
6: Solve problem (8) using $\boldsymbol{X}_{N,s}^{t,*}$, C_s^* to estimate bids $b^{\tau}_{\{up,down\}}$

After receiving the actual dispatch D^{τ} for the current timeslot from the TSO, the aggregator decides upon X_N^{τ} and $\{x_n^{\tau}\}_{n \in N}$ by greedily maximizing the profit in the current timeslot, assuming that its dispatch for future timeslots $t > \tau$ will be $D^t = \mathrm{P}_N^t$:

$$\begin{aligned} \max_{\boldsymbol{x}_n, \boldsymbol{X}_N} & \left\{ \sum_{t \in T} \pi^t \right\} \\ \text{s.t. } & D^t = \mathrm{P}_N^t, && \forall t \in T : t > \tau \\ & \lambda_{\{\mathrm{up,down}\}}^t = 0, && \forall t \in T : t > \tau \\ & x_n^t = \tilde{x}_n^t, && \forall t < \tau, n \in N \\ & (1), (2) \end{aligned} \tag{9}$$

where $\tilde{x}_n^t$ denotes the decisions made in the previous timeslots (which have to be fixed).

3.4 Machine Learning (ML) based algorithmic solution

The FlexOffer estimation part of the method described in the previous section is computationally expensive. That is because set S is exponentially large in the number of timeslots of the horizon T. Thus, many scenarios are needed, where for each scenario the aggregator solves an optimization problem (namely (3.7)). Thus, in real-time operation, there is no sufficient time to apply the method of the previous section.

In order to solve this problem, we propose the use of Machine Learning (ML) techniques to train a decision making system for the aggregator's FlexOffers. We assume that the aggregator knows the form of functions $c_n(\cdot)$ and constraints F_n and has statistical knowledge over their parameters in the form of probability distributions to which these parameters abide. The input data of the ML algorithm, denoted as $\mathcal{U}$, contains all the parameters necessary for defining $\mathrm{P}_N, c_n(\mathbf{x}_n), F_n$. We run Monte Carlo simulations to obtain a set K of samples, where each sample $k \in K$ contains a particular instance $\mathcal{U}_k$ of the input data. For each sample $\mathcal{U}_k$, we apply the method described in the previous section to obtain the estimated bids $b_{k,\{up,down\}}^{\tau}$. Thus, using $\mathcal{U}_k$ and $b_{k,\{up,down\}}^{\tau}$ as input and output respectively, we can train a ML algorithm. The training procedure is summarized in Algorithm 2 below. Once trained, the ML algorithm will be able to provide a fast decision on co-efficients $b_{\{up,down\}}^{\tau}$ for the next timeslot ahead, upon receiving the information on $\mathrm{P}_N, c_n(\mathbf{x}_n), F_n$ in online operation.

Algorithm 2 ML training using Monte Carlo simulations

1: Generate input data $\{\mathcal{U}_k\}_{k \in K}$
2: **for** $k \in K$
3: execute Algorithm 1
4: store $b_{k,\{up,down\}}^{\tau}$
5: Train the ML algorithm using $\mathcal{U}_k$ and $b_{k,\{up,down\}}^{\tau}$

The task at hand is a regression problem. That is, given a specific input, a set of numerical values are predicted. Various ML algorithms were tested for this UCS 4.3 problem. In this document, we present the two methods that achieved the most promising results, namely, Deep Neural Networks (DNNs) and Random Forests (RFs).

3.4.1 Deep Neural Networks (DNNs)

Deep Neural Networks (DNNs) consist of one input layer through which the features are fed into the network. A number of hidden layers follows, each one comprised of several neurons. The large number of layers in DNNs allows the network to learn complex representations. The challenge is to define the number of hidden layers and neurons in order to balance the accuracy and the computational complexity of the model. There is no standard formula to do this, and a trial-and-error approach is usually required.

3.4.2 Random Forests (RF)

Random Forests is an ensemble learning method. Ensemble methods use many learning algorithms combined. They obtain better predictive performance when compared to any of the learning algorithms alone. One ensemble method is bagging of classification or regression trees. In this method, successive trees are independently constructed using a bootstrap sample of the data set. A majority vote is taken for the final prediction. Bagging improves the accuracy and also reduces variance and over-fitting. In random forests, the best split of a given node is decided using a predictor chosen randomly from the set of predictors of that node. Depending on the specific scenario, they can outperform other regression or classification techniques based on support vector machines or neural networks. More information about random forests can be found in 17.

3.5 Simulation setup and performance evaluation results

3.5.1 Simulation setup and evaluation framework

To evaluate the proposed method, we consider a setting where the aggregator represents a portfolio of 100 flexible loads and a RES generation facility. The method is evaluated for an operational horizon of 24 timeslots. A load $n \in N$ features an arrival time arr_n and a departure time dep_n. Its feasible interval for energy allocation is denoted as $H_n = [\mathrm{arr}_n, \mathrm{dep}_n] \subset T$.

The portfolio consists of two classes of loads, namely Thermostatically Controlled Loads (TCLs) $j \in N_{TCL}$, including Air-Conditioners, Water Heaters etc., and EVs $i \in N_{EV}$, where $|N_{TCL}| = |N_{EV}| = 50$ and $N = N_{TCL} \cup N_{EV}$. For each family of loads, we present the models below.

An EV $i \in N_{EV}$ is constrained by an upper and lower power consumption level:

$$\mathrm{x}_i^{\min} \leq x_i^t \leq \mathrm{x}_i^{\max} \qquad (3.10)$$

and it cannot be charged before arrival or after departure:

$$x_i^t = 0, t \notin H_i \qquad (3.11)$$

Moreover, the EV has a certain energy requirement E_i to be fulfilled. When the total charged energy upon departure is less than E_i, the end user bears a cost:

$$c_{EV}(\mathbf{x}_i) = \begin{cases} 0, & \left|\sum_{t\in T} \mathrm{h}_i \cdot x_i^t - \mathrm{E}_i\right| \le \mathrm{tol}_i \\ \mathrm{w}_i \cdot \left(\sum_{t\in H_i} \mathrm{h}_i \cdot x_i^t - \mathrm{E}_i\right)^2, & \left|\sum_{t\in T} \mathrm{h}_i \cdot x_i^t - \mathrm{E}_i\right| > \mathrm{tol}_i \end{cases} \quad (3.12)$$

where tol_i is a tolerance level, h_i is the EV's charging efficiency, and w_i is the load's elasticity parameter. Observe that the EV's cost function exhibits inter-temporal couplings, since the cost of the EV is only realized at its departure timeslot dep_i, but is, however, dependent on the charging decisions of all previous timeslots.

For TCL $j \in N_{TCL}$ let θ_j^t denote the temperature measured by the TCL's sensor. The transition function of the temperature is defined as:

$$\theta_j^t = \theta_j^{t-1} + \mathrm{ins}_j(\theta_{\mathrm{env}}^t - \theta_j^{t-1}) - \mathrm{con}_j x_j^{t-1} \quad (3.13)$$

where θ_{env}^t is the environment's temperature, ins_j is a parameter related to temperature decay (e.g. insulation) and con_j is a conversion factor (from electrical power to thermal energy). Similarly to constraints (3.10) and (3.11), for TCLs we have:

$$\mathrm{x}_j^{\min} \le x_j^t \le \mathrm{x}_j^{\min} \quad (3.14)$$

$$x_j^t = 0, t \notin H_j \quad (3.15)$$

where H_j is the TCL's operation interval. The TCL has a setpoint $\theta_{\mathrm{sp},j}^t$, which represents the user's target temperature. Similarly to EVs, the TCL's cost function is defined as:

$$c_{TCL}(\mathbf{x}_j) = \begin{cases} 0, & \left|\theta_j^t - \theta_{\mathrm{sp},j}^t\right| \le \mathrm{tol}_j \\ \sum_{t\in[\mathrm{arr}_j,\mathrm{dep}_j]} \mathrm{w}_j \cdot \left(\theta_j^t - \theta_{\mathrm{sp},j}^t\right)^2, & \left|\theta_j^t - \theta_{\mathrm{sp},j}^t\right| > \mathrm{tol}_j \end{cases} \quad (3.16)$$

The aggregator also features local RES generation facilities, with a generation profile $\mathrm{R} = \{\mathrm{R}^1, \mathrm{R}^2, \ldots \mathrm{R}^{|T|}\}$. In order to obtain realistic values for P_N, each FlexAsset's intended demand p_n^t for each timeslot, is set to the value that incurs minimum cost to the FlexAsset (assuming no balancing actions by the aggregator), i.e.

$$\mathrm{p}_n^t = \mathrm{argmin}_{x_n^t}\{c_n(x_n)\} \quad \text{s.t. (10)} - \text{(16)} \quad (17)$$

Thus, the Aggregator's net demand profile P_N under no flexibility actions is defined by:

$$\mathrm{P}_N^t = \sum_{n\in N} \mathrm{p}_n^t - \mathrm{R}^t, \quad \forall t \in T \quad (18)$$

Having defined parameters P_N, cost functions $c_n(\mathbf{x}_n)$ and the feasible sets for $\mathbf{x}_n$, we can now apply the ML-based methods proposed in the previous section. Specifically, parameters $\theta_{\mathrm{env}}^t, \mathrm{R}^t, \mathrm{arr}_n^t, \mathrm{dep}_n^t, \mathrm{E}_i, \theta_{\mathrm{sp},j}^t, \mathrm{w}_n$ are the features together with P_N.

Now, we evaluate the proposed ML-based method for the case study presented above. The EVs' charging efficiency, h_i, follows a uniform distribution between 94% and 100%, while the parameter con_j is uniformly sampled from the interval [3, 4]. The average outdoor temperature $\theta_{\mathrm{env}}(average)$ is assumed to follow the temperature of a typical summer day in southern Europe: $\theta^0_{\mathrm{env}}(average) = 83$ F and $\theta^t_{\mathrm{env}}(average) = \theta^{t-1}_{\mathrm{env}}(average) + 3$ F, assuming a simulation horizon of 24 timeslots, that represents quarterly intervals from morning to noon. The actual value for θ^t_{env} follows a normal distribution around the respective value of $\theta^t_{\mathrm{env}}(average)$, with a standard deviation of 3 F. The local RES production for each timeslot is sampled from a normal distribution with mean values starting from 2 kWh and increasing by 2 kWh in every next timeslot. The standard deviation for each timeslot is set equal to 0.25 kWh of the respective mean value. The quantities $\mathrm{B}^t_{\mathrm{up}}, \mathrm{B}^t_{\mathrm{down}}$ were set equal to $0.1\mathrm{P}^t_N$. The rest of the simulation setup's parameters are sampled from normal distributions, as those are defined in the table below. Finally, upon solving the optimization problem of the method, the absolute values were linearized by using an auxiliary variable.

Table 4: Summary of values/distributions of simulation setup's parameters

Parameter	Comments	Value	Average Value	Standard deviation
$\mathrm{x}^{\min}_n$	$\forall n$	0	-	-
$\mathrm{x}^{\max}_i$	for EVs	-	3	0.1
$\mathrm{x}^{\max}_j$	for TCLs	-	5	0.5
arr_i	for EVs	-	4	2.5
arr_j	for TCLs	-	3	1
dep_i	for EVs	-	$\mathrm{arr}_n + 4$	1
dep_j	for TCLs	$\lvert T \rvert$	-	-
E_i	-	-	$\mathrm{x}^{\max}_i(\mathrm{dep}_i - \mathrm{arr}_i) - 2$	0.5
ins_j	-	-	0.05	0.01
$\theta^t_{\mathrm{sp},j}$	-	-	77	1
w_n	$\forall n$	-	0.5	0.1

3.5.1.1 Wholesale Energy Market Model (WEMM)

In order to evaluate the proposed method, we use a model through which the wholesale electricity market receives the FlexOffer of the aggregator and decides whether it is going to request balancing energy (up or down) from the aggregator. In reality, this decision is made by problem (4), where the FlexOffers from all market participants are taken into account. However, since we are only interested in the aggregator's dispatch, for the scope of the FLEXGRID UCS 4.3, we abstract away the complete market model and construct a Wholesale Energy Market Module (WEMM) that provides decisions only on the aggregator's dispatch and the balancing energy price λ^τ for the current timeslot τ.

In case the aggregator is called to offer balancing energy up (i.e. reduce load), it follows that price λ^τ is higher than its offer b^τ_{up}. In this case, the WEMM randomly generates a price that is within the interval $[b^\tau_{up}, \lambda_{max}]$, where λ_{max} is the administrative upper bound for the balancing energy price. In case the aggregator is called to buy balancing energy down (i.e.

increase load), it follows that price λ^τ is lower than its offer b^τ_{down}. In this case, the WEMM randomly generates a price that is within the interval $[\lambda_{min}, b^\tau_{down}]$, where λ_{min} is the administrative lower bound. Parameters $\lambda_{min}, \lambda_{max}$ are set to zero and 20 cents respectively. The model for the WEMM is described in the following procedure:

1. First, the WEMM receives the aggregator's FlexOffers b^τ_{up} and b^τ_{down} for the timeslot ahead.
2. The module randomly decides if it is going to need upward or downward balancing energy, with equal probability unless stated otherwise.
3.
    a. for upward balancing energy (the aggregator reduces its load): The aggregator is not called, (i.e., requested to follow its market schedule, $D^\tau = \mathrm{P}^\tau_N$) with probability $\varrho_{up,out} = max\{1, b^\tau_{up}/\lambda_{max}\}$. On the other hand, the aggregator is called to offer balancing energy $\mathrm{B}^\tau_{\mathrm{up}}$, i.e. $D^\tau = \mathrm{P}^\tau_N - \mathrm{B}^\tau_{\mathrm{up}}$, with probability $\varrho_{up,in} = 1 - \varrho_{up,out}$, at a price λ^τ_{up}, which is picked randomly from the interval $[b^\tau_{up}, \lambda_{max}]$.
    b. for downward balancing energy (the aggregator increases its load): The aggregator is called to offer balancing energy down $\mathrm{B}^\tau_{\mathrm{down}}$ (i.e., $D^\tau = \mathrm{P}^\tau_N + \mathrm{B}^\tau_{\mathrm{down}}$) with probability $\varrho_{down,in} = max\{1, \lambda_{min}/b^\tau_{down}\}$, at a price λ^τ_{down}, which is picked randomly from the interval $[\lambda_{min}, b^\tau_{down}]$. Finally, the aggregator is not called, (i.e., requested to follow its market schedule $D^\tau = \mathrm{P}^\tau_N$) with probability $\varrho_{down,out} = 1 - \varrho_{down,in}$.

The procedure through which the setup is simulated is described in Algorithm 3 below:

Algorithm 3 Evaluation Procedure

1: set $\tau = 1$
2: **while** $\tau \in [1, 24]$
3: feed input features the ML and get ML estimation for $b^\tau_{up}, b^\tau_{down}$
4: feed $b^\tau_{up}, b^\tau_{down}$ to the WEMM and get the dispatch D^τ and price λ^τ
5: solve problem (9) to decide $\{x^\tau_n\}_{n\in N}, X^\tau_N$
6: set $\widetilde{x}^\tau_n$ equal to the solution of (9)
7: $\tau = \tau + 1$

3.5.1.2 Machine learning methods

We assumed 1000 samples to generate a single case of mapping from the defined set of features to the optimal bids $b^\tau_{up}, b^\tau_{down}$. We evaluated the ML algorithms for a total of 1000 cases. We considered 2-fold cross validation with 3 repeats. The accuracy metric is the mean absolute error and the standard deviation of the errors.

Regarding the Deep Neural Networks (DNN) method, we used the Keras deep learning library along with tensorflow. The architecture that we considered is the following: an input layer, seven hidden layers and one output layer. The number of neurons of the input layer and of the hidden layers is equal to the number of features. In the hidden layers, we assumed a dropout rate equal to 0.2. The number of neurons of the output layer is equal to the number

of coefficients $\boldsymbol{b_{up}^{\tau}}, \boldsymbol{b_{down}^{\tau}}$. We chose the rectifier activation function for the hidden layers and the Adam optimization algorithm. Finally, we considered 1000 epochs.

As of the Random Forests (RF) method, we used the Random Forests regressor from the Scikit-Learn library. The number of trees in the forest is 100. The nodes are expanded until all leaves are pure or until all leaves contain less than 2 samples. The minimum number of samples required to be at a leaf node is 1.

3.5.2 Performance evaluation results

In the next subsections, we perform various experiments and simulations with respect to various Key Performance Indicators (KPIs):

3.5.2.1 Comparison of the proposed ML methods

In the table below, we present the mean and the standard deviation of the score (defined as the Mean Absolute Error - MAE) of the ML algorithms for the aforementioned scenarios. We notice that both algorithms are sufficiently accurate even with a relatively low amount of training cases. RFs perform a bit better than DNNs, but the difference is not very large to be deemed significant. Once the data is generated, DNNs require a training time of 60 seconds on a Quad Core CPU at 4 GHz, while RFs require 10 seconds. Note that these running times refer to the training phase. The resulting estimations require much less time (around 0.11 seconds) to provide a bid estimate. Thus, both models are suitable for dynamic scenarios to promptly acquire an efficient FlexOffer decision.

Table 5: Accuracy of ML Algorithms

Algorithm	Mean	Standard Deviation
DNN	0.27	0.005
RF	0.29	0.004

3.5.2.2 Aggregator's profits

Algorithm 3 was run for a number of different cases for the imbalance price λ_{Imb}. More specifically, the setting was simulated for $\lambda_{Imb} = \{0,5,10,\ldots,40\}$. For each value of λ_{Imb}, a number of setting instances were simulated and the results on the aggregator's profits were averaged out over all instances. The figure below shows the resulting average aggregator's profits as a function of λ_{Imb}. The aggregator's profits are always positive. This is not trivial, since if the aggregator does not submit FlexOffers in the balancing market and follows its day-ahead market schedule, it obviously makes zero profit, and if the bidding method performed poorly (e.g. resulted in major imbalances or FlexAsset costs), the aggregator's profit could even be negative.

As it can be observed, the aggregator's profits decline for higher values of λ_{Imb}. However, the curve gradually stabilizes, especially after λ_{Imb} surpasses λ_{max}, which means that as λ_{Imb} increases, the profits are no longer affected significantly by λ_{Imb}. The reason for this, is

that for $\lambda_{Imb} > \lambda_{max}$, the aggregator opts for minimizing its imbalances. Thus, the fact that the profits are not affected by λ_{Imb} after a certain point, means that the aggregator succeeds in minimizing imbalances, which in turn indicates that the proposed ML method achieves a very good capturing of the aggregator's flexibility cost, i.e., the FlexOffers made by the proposed ML method do not result in dispatch decisions that the aggregator cannot eventually follow.

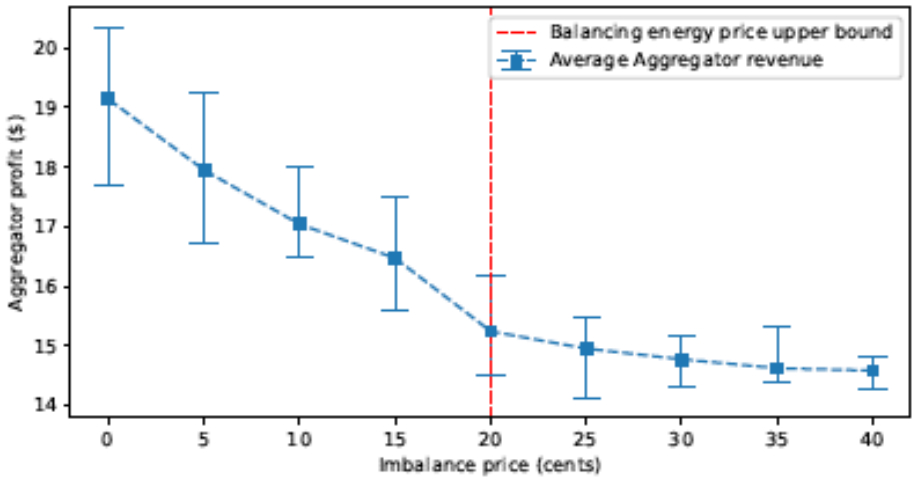

Figure 6: Aggregator's profit as a function of the imbalance price

3.5.2.3 Imbalances

In order to verify the indication of the previous subsection and further elaborate on the previous results, we estimated the probability that the aggregator's FlexOffer results in a dispatch order, which the aggregator does not prefer to follow. This can happen when the flexibility costs of the FlexAssets for following the dispatch, are higher than the imbalance price (which, in turn, means that the estimate of flexibility costs by the ML algorithm was not good). The setting was simulated for different values of parameter tol_n (the same for all FlexAssets). For each value of tol_n, we conducted a number of 1000 simulations and counted the number of experiments in which an imbalance occurred (no matter how small). The probability of imbalance was estimated as the number of experiments with imbalances, divided by 1000, and is depicted in the figure below for different values of tol_n.

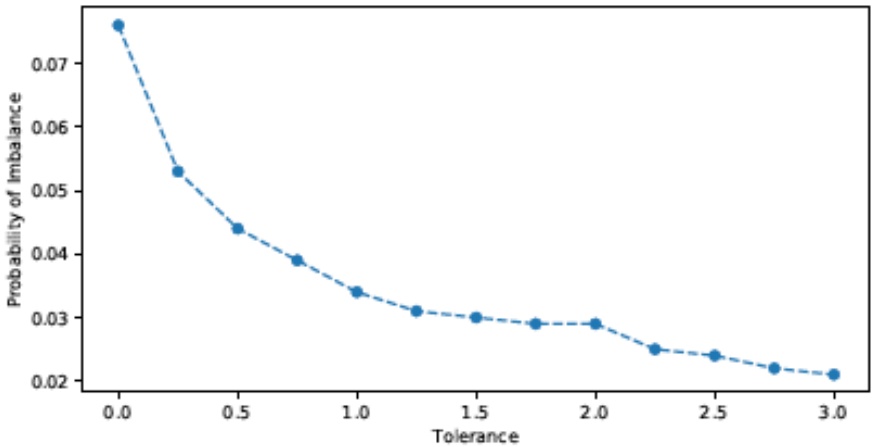

Figure 7: Estimated probability of imbalance for different values of the tolerance level tol_n

3.5.2.4 Flexibility aggregation

In this subsection, we examine how well the proposed flexibility aggregation algorithm captures the FlexAssets' flexibility level. In order to control the overall flexibility of the FlexAssets via a single parameter, we use the tolerance tol_n. A lower value of tol_n means lower flexibility for the set of FlexAssets, since their cost functions (12) and (16) are activated more easily. In contrast, a high tol_n, gives the aggregator more flexibility to shape the FlexAssets' profiles without suffering flexibility costs. The figure below presents the resulted aggregator's FlexOffers (averaged over all timeslots) for different values of tol_n. The figure verifies that for higher values of tol_n, the method is able to capture the increased flexibility of the FlexAssets, since it results in lower FlexOffer. Note that a lower FlexOffer means that the aggregator is more likely to be dispatched (even for a lower price), therefore it communicates to the TSO that the aggregator is more flexible towards offering balancing energy.

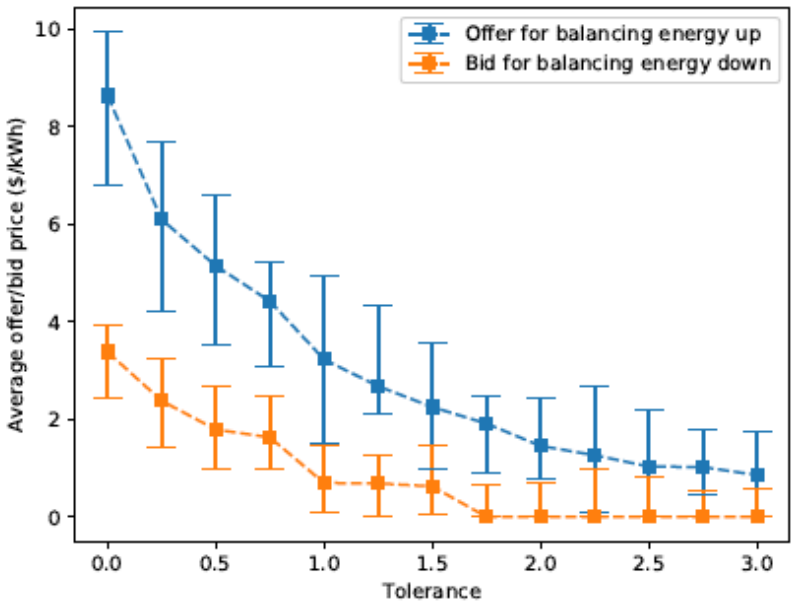

Figure 8: Average aggregator's offers/bids for different levels of FlexAsset flexibility

3.5.2.5 FlexOffer behavior

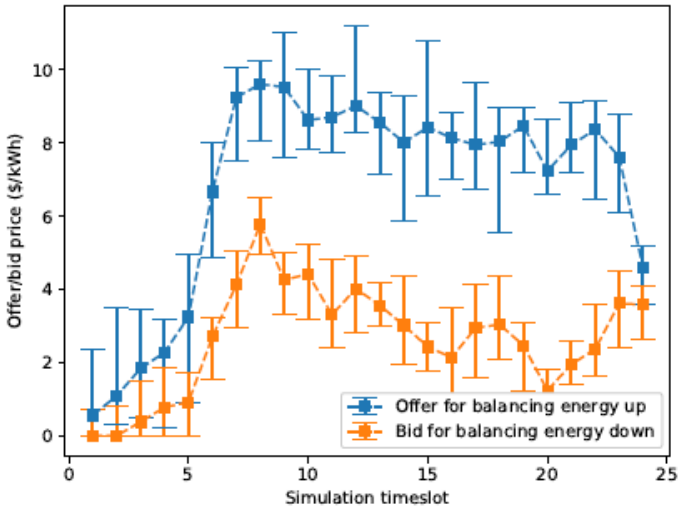

Figure 9: Aggregator's offers/bids for each timeslot

For the purposes of this experiment, we modified the WEMM to always ask from the aggregator to offer balancing energy up (curtail load). Thus, the aggregator curtails energy consumption and in every next timeslot, it is asked to curtail again. The figure above shows how the aggregator's FlexOffers b_{up}^{t}, b_{down}^{t} are affected in this case along the time horizon. The results indicate that when the aggregator is asked to curtail energy in a given timeslot, the proposed method increases the requested price for further curtailment in the next timeslot and, after a certain point, increases also the aggregator's FlexOffer to buy balancing energy. After a certain point in time, this phenomenon is counter-balanced by the departure of many EVs (and the arrival of new ones), which is why the offer/bid prices do not further increase after that point.

3.6 Next research steps for the M19-M26 period

Within M19-M26, we will elaborate on the UCS 4.3 work in order to convey more system-level simulations to evaluate the performance of our proposed algorithms for the Distribution Level Flexibility Markets (DLFM), too. As already mentioned above, in this report we decided to focus on the TSO's balancing energy market, which represents the existing regulatory conditions in most EU countries, today. In the next months, we will focus on the aggregator's participation in near-real-time DLFM.

This work is closely inter-related with UCS 2.3, which refers to ESP's stacked revenue maximization by its participation in several markets, such as: i) day-ahead energy market operated by MO, ii) day-ahead reserve market operated by TSO, iii) day-ahead DLFM operated by FMO, iv) near-real-time DLFM operated by DSO, and v) near-real-time balancing market operated by TSO. Therefore, we will provide more performance evaluation results showcasing the optimal FlexOffers made by an aggregator in order to co-optimize its participation in all the above-mentioned markets.

Another research task, which is also related with respective WP6 work is to integrate the proposed FlexOffer creation algorithm into the Automated Flexibility Aggregation Toolkit (AFAT) and FLEXGRID ATP. Thus, the aggregator user will be able to utilize the AFAT to make efficient FlexOffers in near-real-time balancing markets and DLFMs. In the online operation mode, the aggregator will be able to automatically create 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). In the offline operation mode, the aggregator will be able to run "what-if" scenarios to see whether it is more beneficial to participate in the existing TN-level balancing market or DN-level balancing market (i.e. DLFM). If the FlexOffer is not accepted in DLFM, it can be automatically forwarded to the TSO's balancing market via a respective Application Programming Interface (API).

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 UCS 4.2. In FLEXGRID, we propose a novel B2C flexibility market architecture through which an aggregator will be able to optimally operate a market in which the various small-scale distributed FlexAssets (DFAs) compete with each other. 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. 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. A main research novelty of our proposed 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.

Within FLEXGRID UCS 4.2 context, we **propose advanced retail market mechanisms (ARMM) that can be used by an aggregator in order to operate a novel B2C flexibility market architecture**. Our ultimate goal is to integrate the most important research algorithms in the FLEXGRID ATP (TRL 5) and more specifically in the Automated Flexibility Aggregation Toolkit (AFAT).

Through AFAT, **the aggregator user 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 which end energy prosumers compete with each other. 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.

Based on the AFAT's results (TRL 5), **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.

4.1 Problem statement, related state-of-the-art and FLEXGRID research contributions

In an environment with high RES penetration, an important asset is the load flexibility at the demand side. More precisely, Distributed Flexibility Assets (DFAs) are distributed ESS and smart devices that exhibit flexibility in their energy demand (e.g. EVs and HVAC units). DFAs are envisaged to participate in electricity markets through energy aggregators. Academic research was quick to design optimization and control methods for extracting the value of DFAs. However, actual end-user engagement and adoption of real-life commercial applications are yet to catch up. A significant barrier has been the lack of intelligent agents that negotiate with aggregators on behalf of the end user and deliver an attractive trade-off between consumption profile and energy bill reduction. However, advanced modelling tools and advancements in digital economies are ready to facilitate real time market interaction between intelligent end user agents and intelligent aggregator agents. In this way, aggregators can buy flexibility from DFAs in order for the former to be able to enhance their position in B2B markets and increase their profitability, while also offering services to the grid operators.

In FLEXGRID, we consider B2B flexibility markets in which the system operators (i.e. DSO/TSO) are FlexBuyers, while aggregators are FlexSuppliers. Moreover, we also assume ad-hoc B2C flexibility markets (or else retail flexibility markets) operated by an aggregator entity, in which end energy prosumers compete with each other. In order to design such B2C flexibility markets in an efficient way, novel Market Mechanisms (MMs) need to be designed. A MM includes the bidding protocols for the market participants and the rules of market operation, namely an allocation rule and a pricing rule. Traditional, static retail pricing schemes are not able to capture the dynamics of the electricity network and thus traditional utilities fail to catch up with the new needs of the energy market. In contrast, dynamic MMs are needed in order to efficiently manage DFAs.

A Market Mechanism (MM) can be: i) **Iterative:** Traditional time-of-use (TOU) MMs are open-loop, in the sense that the end user is insulated from the other users' actions (this lack of feedback and coordination between users is a major source of instability) and ii) **Online:** There is inherent uncertainty in the electricity grid, partly because of imperfect forecasts about the inflexible part of user demand and partly because of uncertainty in the supply side. A dynamic MM needs to be able to adjust to online signals in real-time. On the other hand, dynamic/intelligent MMs need a corresponding intelligence on the end user side in order to function. For example, an end user that is manually making decisions cannot catch up with an iterative and online MM. This means that intelligent agents have to be developed on the end user's side, so as to negotiate on the user's behalf.

From a research perspective, **Market Mechanisms (MMs)** for electricity grids can be generally evaluated through seven requirements (KPIs):

1. **Optimality/efficiency**: The difference between the value that users give to their Energy Consumption Curves (often called utility function) and their energy cost (this difference is noted in the international literature as Social Welfare).

2. **Incentive Guarantees/Strategy proof**: The resilience of the system to end users who benefit from declaring false preferences.
3. **Privacy protecting**: The quantity of information that is required from the end user.
4. **Convergence/scalability**: The speed of convergence of MMs and its scalability with respect to the number of end users.
5. **Fairness**: The policy for distributing the energy costs and awards to energy consumers should be fair so that it is also able to trigger behavioural changes (e.g. participation in flexibility markets). For example, end users would be unwilling to participate to Demand Response events if not-participating or cheating end users benefit from them.
6. **Competitiveness/sustainability**: The MM should offer to the end users (prosumers) attractive charges with respect to their Energy Consumption Curve (ECC), while being practically implementable/realizable.
7. **Externalities**: Other positive/negative outcomes of the MM (e.g., controllability in order to satisfy system-wide constraints, simplicity for end users to understand the mechanism, etc.).

A detailed survey on related works from the international literature is provided in section 5.2 of the previous D3.1[13], so the interested reader can refer to this document for extensive details. This survey work identified all related research ideas for deploying advanced retail pricing schemes and market mechanisms in the modern electricity markets. Here, we summarize the FLEXGRID's contributions and novelties.

Within the FLEXGRID context, we propose a **novel B2C flexibility market operated by an aggregator**, which deals with all the afore-mentioned requirements (or else KPIs) as follows:

1. Regarding **"optimality/efficiency" KPI**, we prove that our proposed scheme achieves the Vickrey-Clarke-Groves (VCG) outcome. VCG mechanism is broadly considered as the cornerstone mechanism design as it is provably the unique mechanism that achieves the optimal social welfare. Thus, our proposed scheme does not sacrifice efficiency at all to satisfy other KPIs.
2. As of **"incentive guarantee/strategy proof" KPI**, our scheme satisfies the Dominant-Strategy-Incentive-Compatibility (DSIC) property, which is considered the strongest incentive guarantee in the international literature.
3. Regarding the **"privacy protecting" KPI**, our proposed scheme is suitable for a distributed implementation unlike the centralized optimization solution that has already been analyzed in UCS 4.1 in chapter 2 of this document. In other words, this means that the end energy prosumers do not have to reveal their utility functions (or else sign any binding FlexContract with the aggregator), but just respond to the retail/B2C flexibility market price signals sent by the aggregator in an online fashion.
4. Regarding the **"convergence/scalability" KPI,** we also prove both analytically and via system-level simulations (cf. section 4.5 below) that our proposed algorithmic solution converges quickly for a large number of end energy prosumers that participate in the B2C flexibility market.

[13] https://flexgrid-project.eu/assets/deliverables/FLEXGRID_D3.1_final_version_29092020.pdf

5. With the term **"fairness"**, we refer to the problem of state-of-the-art Real Time Pricing (RTP) schemes, which do not strongly motivate end energy prosumers to modify their electricity consumption habits. This happens mainly due to the fact that in each given timeslot, all end users get the same real-time price, namely both flexible and inflexible end users will get the same reward for this given timeslot. Our proposed market mechanism is based on our previously published work on a Behavioral RTP (B-RTP) scheme[14] that offers an easily adjustable level of financial incentives to end users by fairly rewarding the desirable behavioral electricity consumption changes.
6. As of **"competitiveness/sustainability" KPI**, via our proposed scheme, the aggregator can configure two basic parameters (i.e. 'γ' and 'p' parameters). 'γ' parameter defines the level of price-based incentives provided to each end user towards behavioral change, while 'p' parameter denotes the percentage of aggregator's revenues that will be distributed to end users, while the residual amount of revenues will be kept as aggregator's profits. Hence, the aggregator can easily adjust these parameters in order to be able to adapt to the ongoing conditions of its business (e.g. when the competition with other aggregators is harsh, then it can keep less profits for itself and distribute more financial rewards to flexible end energy prosumers).
7. Finally, as an **"externality" KPI**, the proposed scheme is transparent to any type of FlexRequest that needs to be satisfied. For example, a FlexRequest can be created by: i) a DSO to deal with local congestion and voltage control issues, ii) a TSO to deal with transmission network level imbalances, iii) the aggregator itself to deal with its internal portfolio's imbalances, iv) a BRP to deal with its imbalances in a given geographical area under its balancing responsibility, etc. This functionality is very important for the proposed B2C flexibility market as it can be easily integrated with the structure and respective needs of the B2B flexibility markets realized in FLEXGRID ATP, too.

4.2 System model

Within FLEXGRID project's context, we consider a novel B2C flexibility market comprised of an aggregator and a set $\mathcal{N} \triangleq \{1,2,\dots,n\}$ of n self-interested end energy prosumers, hereinafter referred to as end users. We also consider a discrete representation of time, where continuous time is divided into timeslots $t \in \mathcal{T}$ of equal durations s, where set $\mathcal{T} \triangleq \{1,2,\dots,m\}$ represents the scheduling horizon. Each end user possesses a number of controllable appliances, with each appliance bearing an energy demand. If the consumptions of different appliances are not coupled (independent of each other), the appliances can participate in the FlexRequest (or else DR event) virtually as different end users. The bills of an end user's appliances will add up to calculate the bill of the actual end user. Thus, throughout this chapter, we can consider one appliance per end user for ease of presentation.

[14] K. Steriotis, G. Tsaousoglou, N. Efthymiopoulos, P. Makris, E. Varvarigos, "A Novel Behavioral Real Time Pricing Scheme for the Active Energy Consumers' Participation in Emerging Flexibility Markets", Elsevier Sustainable Energy, Grids and Networks (SEGAN) Journal, vol. 16, pp. 14-27, Dec 2018, https://www.sciencedirect.com/science/article/pii/S2352467718300201.

4.2.1 End user's energy consumption model and utility function

An appliance requires an amount of energy for operation. For example, if an appliance's operating power is 1Watt, and $s = 1$ hour, then the energy that the appliance consumes in one timeslot of operation is $1Wh$. This energy consumption is measurable in real-time and can be shed upon request, in exchange for monetary compensation. Such a request for consumption modification may be called a FlexRequest that is published in a S/W platform or else online marketplace like the FLEXGRID ATP that is developed within FLEXGRID WP6 context. In cases where a FlexRequest is scheduled ahead of time, a strategic end user can falsely claim that he/she intended to consume energy at the time of the FlexRequest and that he/she "reduced" consumption in response to the FlexRequest, while in reality he/she never intended to consume energy in that time. However, in this work, we deal with a real-time DR process. Each end user's consumption is measured in real-time, and, when a FlexRequest occurs, the end user's curtailment is measured against his/her last real-time consumption measurement and not against the end user's declared intended consumption. Thus, the consumption measurement is taken before the FlexRequest, so the end user cannot manipulate it since he/she does not know when a FlexRequest is going to occur. The high-level system model that we consider in this UCS is illustrated in the figure below.

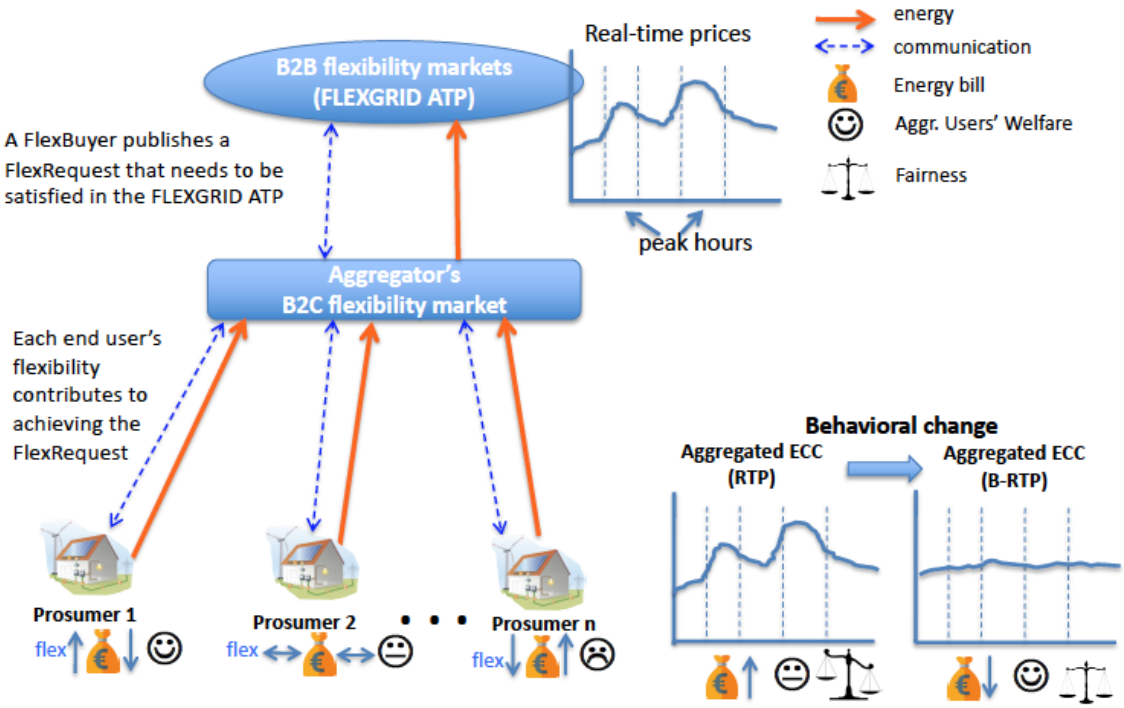

Figure 10: System model for FLEXGRID UCS 4.2

A request for consumption reduction (i.e. FlexRequest) is made by some third party (e.g. the DSO/TSO) along with the respective reward function (i.e. price/quantity curve) and the aggregator takes on the task of providing the requested service by setting up a B2C flexibility market among the end energy prosumers of its portfolio. Upon a FlexRequest in timeslot t, the aggregator offers a per-unit reward to the end users for consumption reduction. User i can respond by reducing his/her consumption by a quantity q_i^t, assumed to be positive ($q_i^t \geq 0$), without loss of generality. As described earlier and also in chapter 5 of previous D3.1, the decisions for q_i^t can be taken by an intelligent agent (on behalf of the actual end user and

according to the end user's preferences) in order to disengage the actual end user from real-time participation.

The consumption reduction q_i^t is characterized by its feasible set Q_i (defined by a set of constraints on q_i^t) and the discomfort function $d_i(q_i^t)$ of end user i. The discomfort function is private to each end user and expresses the minimum compensation in monetary units (euros) that an end user requires, in order to reduce his/her consumption by the corresponding amount. The discomfort, as a function of q_i^t, can take various forms depending on the appliance. We make the following assumptions on the form of function $d_i(q_i^t)$:

Assumption 1. Zero consumption reduction, brings zero discomfort to the user: $d_i(0) = 0$

Assumption 2: The discomfort function is convex, so that additional increase of q_i^t brings increasing discomfort to the user:

$$q_{iA}^t \geq q_{iB}^t \;\Leftrightarrow\; d_i(q_{iA}^t + \varepsilon) - d_i(q_{iB}^t + \varepsilon) \geq d_i(q_{iA}^t) - d_i(q_{iB}^t), \forall \varepsilon, q_{iA}^t, q_{iB}^t > 0.$$

Detailed example appliance models (including operational constraints) are described in section 4.5 below. Nevertheless, the discomfort function $d_i(q_i^t)$ is kept general for the moment in order to emphasize that the theoretical results to be presented in the following sections are valid for any end user model that satisfies the assumptions above.

In order to incentivize end users towards reducing their consumption, the aggregator offers a reward $r_i(q_i^t)$. An end user's utility is defined as the difference between his/her discomfort for the consumption reduction realized and the reward he/she receives for achieving this reduction:

$$U_i = \sum_{t \in \mathcal{T}} [r_i(q_i^t) - d_i(q_i^t)] \,. \qquad (4.1)$$

In order to offer the rewards $r_i(q_i^t)$, the aggregator draws on the reward offered by the FlexBuyer that requests the reduction (i.e. FlexRequest), as described in the following subsection.

4.2.2 FlexRequest and the aggregator's problem

Let L^t denote the aggregated consumption of all end users in $\mathcal{N}$, as seen by the FlexBuyer, within a certain time interval t. The energy cost is modeled as a quadratic function of L^t:

$$C^t = c_1 L^t + c_2 (L^t)^2$$

Upon a new FlexRequest, the FlexBuyer asks for a reduction of the end users' aggregated consumption and offers monetary incentives to the aggregator towards its realization. Let D^t denote the reduction from baseline consumption L_B^t to consumption $L_B^t - D$ at time interval t. The respective cost reduction is:

$$C(L_B^t) - C(L_B^t - D) = c_1 L_B^t + c_2 (L_B^t)^2 - c_1 (L_B^t - D) - c_2 (L_B^t - D)^2$$

which reads:

$$C(L_B^t) - C(L_B^t - D) = (c_1 + 2 * c_2 * L_B^t) * D - c_2 D^2$$

We set $a = c_1 + 2 * c_2 * L_B^t$ and $b = c_2$ and the cost benefit $C(L_B^t) - C(L_B^t - D)$ is denoted as a reward function $R(D)$:

$$R^t(D^t) = a \cdot D^t - b \cdot (D^t)^2, \;\; D^t \in [0, L^t], \qquad (4.2)$$

where a, b are positive parameters with $a \geq 2bL^t$ so that it is an increasing function in the range of permitted values. The proposed B2C flexibility market architecture is open to any other choice of $R^t(D^t)$, provided it is an increasing and concave function. Thus, we assume that upon a FlexRequest, the FlexBuyer offers a marginal per-unit reward for a reduction of D^t units.

$$\mu = \frac{d\left(R^t(D^t)\right)}{d(D^t)} \qquad (4.3)$$

The aggregator is responsible for aggregating the end users' participation in the FlexRequest, coordinating their actions, and dividing the compensation profits (rewards) among the end users. We assume a communication network, built on top of the electricity grid, through which the aggregator can exchange messages with the end users in an iterative and online fashion as already explained above.

4.3 Problem Formulation

With respect to the system model described above, we would like to facilitate the allocation of consumption reduction among the end users to maximize social welfare. Social welfare is defined as the difference between the revenues $R^t(D^t)$ that the aggregator receives from the FlexBuyer for the consumption curtailment D^t, as defined in Eq. (4.2), and the sum of the discomforts that this curtailment causes to its end users. This problem can be formulated from Eqs. (4) and (5) below:

$$\max_{q_i^t \in \mathrm{Q}_i, i \in \mathcal{N}} \{R^t(D^t) - \textstyle\sum_{i \in \mathcal{N}} [d_i(q_i^t)]\} \qquad (4.4)$$

$$s.t. \;\; D^t = \textstyle\sum_{i \in \mathcal{N}} q_i^t \qquad (4.5)$$

The problem defined by Eqs. (4.4) and (4.5) is a convex optimization problem and could be solved efficiently if the local functions $d_i(q_i^t)$ were known or truthfully disclosed by all end users. However, $d_i(q_i^t)$ of each user is not known and thus, problem (4.4) is typically solved via dual decomposition in the demand side management (DSM) literature (see more details about state-of-the-art related works in section 5.2 of D3.1). In this approach, the aggregator iteratively increases a per-unit reward λ asking from the end users their consumption reduction $q_i^t(\lambda)$ at each per-unit reward λ (auction query). At each iteration, each end user *i* responds with his/her preferred $q_i^t(\lambda)$. A truthful (locally optimal) response by end user i, denoted as $\widetilde{q_i^t}(\lambda)$, is one that maximizes i's utility for reward λ. This is mathematically formulated as the solution to maximization problem (4.6):

$$\widetilde{q_i^t}(\lambda) = \underset{q_i^t \in Q_i, i \in \mathcal{N}}{\operatorname{argmax}} \{\lambda \cdot q_i^t - d_i(q_i^t)\} \quad (4.6)$$

Clearly, $\widetilde{q_i^t}(\lambda)$ is non-decreasing in λ, since $q_i^t \geq 0$. The auction terminates when λ reaches a value for which $\sum_{i \in \mathcal{N}} q_i^t(\lambda) = D^t(\lambda)$. The final price is called the market-clearing price of the B2C flexibility market and is denoted by λ_{mc}. The allocation at λ_{mc} is efficient if the end users truthfully report their q_i^t at each query. However, truthful report may not be the best strategy for every end user. To illustrate this, we present the following example.

Consider two end users and a given timeslot t. End User 1 operates a load with power consumption 10 kW, while end user 2 operates a 50 kW load. Now suppose they participate in a FlexRequest and their discomfort function is $d_i(q_i^t) = \omega_i \cdot (q_i^t)^2$, $i \in \{1,2\}$, where their true flexibility parameters are $\omega_1 = \omega_2 = 0.1$. The reward function is $R^t(\Delta L^t) = 5 \cdot (\Delta L^t)$. Should they act according to their true discomfort function parameters, their utilities (given by Eq. (4.1)) at equilibrium would be $U_1 = U_2 = 4.875$ units. In case end user 2 acts untruthfully according to $\omega_2^{fake} = 0.2$, his/her utility at equilibrium will be $U_2 = 7$. Therefore, the best strategy for User 2 is to be untruthful.

The previous example demonstrates how the market-clearing approach builds on the assumption that end users behave myopically, by truthfully solving (4.6) at each iteration. However, a FlexRequest will involve intelligent agents and it will not take long before such end users realize that they can benefit from engineering untruthful responses. The problem is that if we relax the truthfulness assumption and consider strategic end users, market-clearing methods no longer result in efficient allocations. **Thus, it is very important to design a market mechanism (MM) that is not only efficient, but also incentive compatible.**

The Vickrey-Clarke-Groves (VCG) mechanism is the unique mechanism that is simultaneously DISC (Dominant Strategy Incentive Compatible) and efficient[15]. The VCG payment rule is the so called "Clarke pivot rule", which calculates a reward r_i equal to i's "externality". In other words, it rewards each end user i with an amount equal to the difference that i's presence makes in the social welfare of other end users. In the direct VCG mechanism, users are asked to declare their local functions $d_i(q_i^t)$ to the aggregator. Because of the Clarke pivot rule, it is a dominant strategy for each end user to make a truthful declaration. Thus, the efficient allocation that corresponds to the social welfare maximization problem can be calculated at the aggregator's side. In order to calculate the VCG rewards from Eq. (4.6), problem (4.4) is solved $|\mathcal{N}| + 1$ times (one time with each end user in $\mathcal{N}$ absent to calculate the payments, plus one time with all end users present to calculate the allocation). The major drawback of the direct VCG mechanism is the requirement that the end users disclose their discomfort functions $d_i(q_i^t)$ to the aggregator. This raises important issues such as a) Lack of privacy, in the case where end users are reluctant to reveal local information, and b) Difficulty of implementation, in cases where end users are unable to express their preferences in a closed form function.

[15] V. Krishna, "Auction Theory", New York: Academic, 2002.

In the next section, we propose a modification of Ausubel's Clinching auction[16], allowing for a distributed implementation of VCG, which is designed to tackle these issues. In particular, we opt for an iterative auction that:

- facilitates end users' bids via auction queries, thus making the proposed architecture more easily implementable in practice
- engages end users in the B2C flexibility market and allocates consumption reduction gradually along the way, so that price discovery is facilitated on the end users' side
- protects end user's privacy via a properly designed communication protocol

4.4 Proposed algorithmic solution

4.4.1 Ausubel's Clinching auction and the proposed Modified Clinching Auction (MCA) algorithm

The Clinching Auction (CA) is a well-known ascending price auction that halts when demand equals supply. However, in contrast to most auctions, allocation and rewards are not cleared exclusively at the final iteration. Rather, the goods (consumption reduction in our context) are progressively allocated as the auction proceeds and payments are also progressively built, while the auction design guarantees that the final allocation and payments coincide with the ones obtained through VCG. Thus, both allocation efficiency and incentive compatibility are achieved, while the aforementioned privacy and implementation drawbacks of the direct-VCG mechanism are effectively addressed.

In order for the Clinching Auction to work in our setting, first we need to reverse the price trajectory. In the proposed Modified Clinching Auction (MCA), the aggregator begins with a per-unit reward $\lambda = \lambda_{max}$ and in each iteration k the price λ^k is reduced by a small positive number ε. The size of ε adjusts the discretization level of MCA. By Eq. (3), reward λ_{max} is $\frac{dR^t(0)}{d\Delta L^t} = a$, which, as analyzed earlier, is the highest value possible given that R^t is concave. End users respond by bidding their preferred reduction $\widetilde{q_i^t}(\lambda)$ for each λ. We represent the end user's response at λ as the solution to the end user's utility maximization problem (which is formally defined in Eq. (4.6)).

The end user's objective function is concave in q_i^t, since $\lambda \cdot q_i^t$ is linearly increasing and $d_i(q_i^t)$ is convex by *Assumption 2*. Also, the solution $\widetilde{q_i^t}$ is increasing in λ, which means that the end user's response $\widetilde{q_i^t}$ gradually decreases as λ decreases. For the MCA, we relax constraint (4b) to the inequality:

$$D^t \geq \sum_{i \in \mathcal{N}} q_i^t \qquad (4.7)$$

Consider an arbitrary iteration k of the MCA and let $D^t(\lambda^k)$ denote the FlexBuyer's desired reduction for per-unit reward λ^k. The central idea of the MCA is the following: if there is a set $\mathcal{N}^j \subset \mathcal{N}$ for which we have:

[16] L. M. Ausubel "An efficient ascending-bid auction for multiple objects", in American Economic Review, vol 94, no.5, pp, 1452–1475, 2004.

$$D^t(\lambda^k) - \sum_{j\in\mathcal{N}^j}\left(\widetilde{q}_j^t(\lambda^k)\right) > 0 \qquad (4.8)$$

then we allocate a reduction equal to $\zeta_i^k = D^t(\lambda^k) - \sum_{j\in\mathcal{N}^j}\left(\widetilde{q}_j^t(\lambda^k)\right)$ to each end user $i \notin \mathcal{N}^j$ at a per-unit reward λ^k. We then say that end user i "clinched" ζ_i^k units. The MCA auction terminates when set $\mathcal{N}^j$ that satisfies condition (4.8) and set $\mathcal{N}$, are equal, that is, constraint (4.7) is satisfied. After that, it allocates the remaining $D^t(\lambda^{k-1})$ proportionally to the end users that bid in the second-to-last iteration.

The critical advantage of the Clinching auction is that it allocates different amounts of flexibility units at different rewards, and the flexibility units that an end user clinches do not depend on his/her own bid, but only on the other end users' bids. The algorithm that implements MCA is presented in the table below.

Table 6: The proposed Modified Clinching Auction (MCA) algorithm

1.	Initialize $\lambda^0 = \lambda_{max}$, $q_i^t(\lambda_{max})$, $D^t(\lambda_{max})$, $k = 0$
2.	**while** $D^t(\lambda^k) < \sum_{i\in\mathcal{N}}\left(\widetilde{q}_i^t(\lambda^k)\right)$
3.	**if** there exists $\mathcal{N}^j$: $\sum_{j\in\mathcal{N}^j}\left(\widetilde{q}_j^t(\lambda^k)\right) < D^t(\lambda^k)$
4.	clinch units $\zeta_i^k = D^t(\lambda^k) - \sum_{j\in\mathcal{N}^j}\left(\widetilde{q}_j^t(\lambda^k)\right)$ for all $i \notin \mathcal{N}^j$ at per-unit reward λ^k
5.	**else**
6.	set $\lambda^{k+1} = \lambda^k - \varepsilon$ and $k = k + 1$
7.	ask each end user a reduction query for λ^k and collect the responses $q_i^t(\lambda^k)$
8.	ask the FlexBuyer for the desired total reduction $D^t(\lambda^k)$ at per-unit-reward λ^k
9.	**End while**
10.	Clinch units $\zeta_i^k = \left(q_i^t(\lambda^{k-1}) - \sum_{\xi=0}^{k-1}\zeta_i^\xi\right)\cdot\frac{D^t(\lambda^{k-1})}{\sum_{i\in\mathcal{N}} q_i^t(\lambda^{k-1})}$ at per-unit reward (λ^{k-1}), for each $i \in \mathcal{N}$

We are now able to prove the optimality of MCA in terms of social welfare performance:

Theorem 1: The social welfare loss at the final allocation of MCA is within $(\varepsilon^2 + \lambda_{max}\cdot\varepsilon)/2b$ of the maximum possible.

Proof: The value of λ at which $D^t = \sum_{i\in\mathcal{N}}\left(\widetilde{q}_i^t\right)$ is denoted as λ_{mc}, which gives

$$D^t(\lambda_{mc}) = \sum_{i\in\mathcal{N}}\left(\widetilde{q}_i^t(\lambda_{mc})\right) \qquad (4.9)$$

Let $\mathcal{k}$ denote the number of iterations until the auction halts, that is,

$$\mathcal{k} = \left\lceil\frac{\lambda_{max}-\lambda_{mc}}{\varepsilon}\right\rceil, \qquad (4.10)$$

where $\lceil\cdot\rceil$, denotes the rounding to the nearest larger integer. We have:

$$\left\lceil \frac{\lambda_{max}-\lambda_{mc}}{\varepsilon} \right\rceil \leq \kappa \leq 1 + \left\lceil \frac{\lambda_{max}-\lambda_{mc}}{\varepsilon} \right\rceil \qquad (4.11)$$

After the last "clinchings" (cf. line 10 of the algorithm), we have efficiently allocated $D^t\left(\lambda^{\kappa-1}\right)$ reduction units to the end users. The remaining $D^t(\lambda_{mc}) - D^t\left(\lambda^{\kappa-1}\right)$ are not allocated and this causes the loss of welfare W_{loss}, which is depicted as the grey area in the figure below, where the red line represents $D^t(\lambda)$ and the blue line represents $\sum_{i\in\mathcal{N}} \widetilde{q_i^t}(\lambda)$.

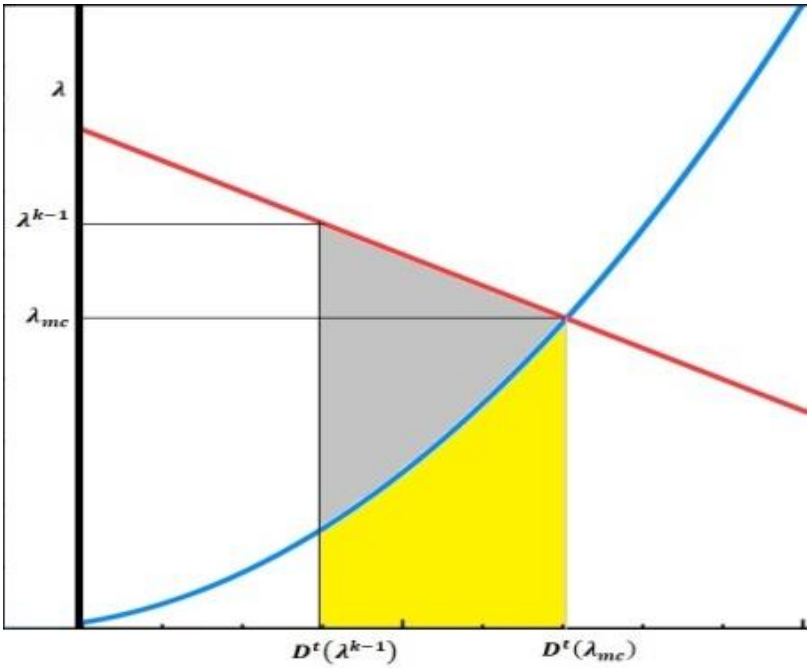

Figure 11: $D^t(\lambda)$ and $\sum_{i\in\mathcal{N}}\left(\widetilde{q_i^t}(\lambda^k)\right)$ as a function of λ

Since we remain agnostic of the closed form of $\sum_{i\in\mathcal{N}}\left(\widetilde{q_i^t}(\lambda^k)\right)$, we assume the worst case and calculate an upper bound on the sum of the grey plus the yellow area of the figure above:

$$W_{loss} \leq \lambda_{mc}\left(D^t(\lambda_{mc}) - D^t\left(\lambda^{\kappa-1}\right)\right) + \frac{1}{2}(\lambda^{\kappa-1} - \lambda_{mc})\left(D^t(\lambda_{mc}) - D^t\left(\lambda^{\kappa-1}\right)\right).$$

By substituting $D^t(\lambda) = \frac{a-\lambda}{2b}$ from Eq. (3), we get:

$$W_{loss} \leq \frac{\lambda_{mc}(\lambda^{\kappa-1}-\lambda_{mc})}{4b} + \frac{\lambda^{\kappa-1}(\lambda^{\kappa-1}-\lambda_{mc})}{4b} \leq \frac{\left(\lambda^{\kappa-1}\right)^2-(\lambda_{mc})^2}{4b}.$$

By further substituting $\lambda^{\kappa-1} = \lambda_{max} - \varepsilon(\kappa - 1)$ and also substituting κ from inequalities (4.11), using the left inequality when κ appears with a minus sign and the right inequality when it appears with a plus sign, we finally obtain:

$$W_{loss} \leq \frac{\varepsilon^2+\lambda_{max}\cdot\varepsilon}{2b},$$

completing thus the proof.

Since we cope with a real-time application, the trade-off between the market mechanism's optimality and its computational time is of special importance. The latter directly relates to the price-step ε, which means that Theorem 1 gives a quantification of the trade-off

described. Thus, the aggregator can accurately predict and control the mechanism's response time by adjusting ε, (e.g. in order to meet the balancing market's time granularity), while having a worst-case quantification of the efficiency loss.

In practice, for the relevant use cases of price-anticipating end users, the computational complexity of the MCA is small, which allows for a very small choice of ε. To emphasize this, it is useful to state the following corollary to Theorem 1.

Corollary 1: for $\varepsilon \ll 1$, the welfare loss grows linearly with ε.

Because the MCA includes a price-sensitive response also at the operator's side, we have to verify that the properties of efficiency and incentive compatibility still hold. This is proved in the following Propositions.

Proposition 1: Truthful bidding is a dominant strategy in MCA.

Proof: Fix an iteration k and assume that user i bids $q_{i,false}^{t}(\lambda^k) \neq \widetilde{q_i^t}(\lambda^k)$ in that iteration. From step 4 of MCA, we see that ζ_i^k does not depend on q_i^t but only on the other end users' bids $q_j^t, j \neq i$. Thus, end user i's bid can affect i's allocation only by changing the λ at which the termination condition holds. This means that a false bid $q_{i,false}^{t}(\lambda^k)$ will make a difference to i, only if k is the last iteration. However, by definition of $\widetilde{q_i^t}(\lambda^k)$ (see Eq. (6)), any bid $q_{i,false}^{t}(\lambda^k) \neq \widetilde{q_i^t}(\lambda^k)$ brings strictly lower utility to end user i at any iteration k. Thus, truthful bidding brings the highest utility to end user i.

Furthermore, the following properties of the VCG mechanism hold also for the MCA:
Proposition 2: MCA is *individually rational*, weakly *budget-balanced*, and *achieves the maximum revenue* for the aggregator among all efficient mechanisms.

Proof: The MCA auction is welfare maximizing (by Theorem 1, for ε small enough) and DSIC (by Proposition 1). Moreover, the class of VCG mechanisms is the unique class that simultaneously achieves these two properties. Since MCA terminates with the VCG allocation and payments, it inherits the property of *individual rationality.*

Regarding the weak budget balance property, it suffices to show that our setting exhibits the no single-agent effect. An environment exhibits no single-agent effect if the aggregated utility of $n-1$ users does not improve by adding a n^{th} user to the system. This property holds in single-sided auctions with monotonous preferences[17], since dropping an end user only reduces the competition for the remaining end users, thus making them better-off.

Moreover, the VCG market mechanism maximizes the auctioneer's utility, which means that the aggregator buys flexibility units from the end users at the lowest possible price (among all efficient and individually rational market mechanisms).

[17] Y. Shoham, K. Leyton-Brown, "Multiagent Systems", Cambridge University Press, 2009.

4.4.2 Privacy preserving distributed communication protocol

A major drawback of the direct VCG mechanism is that it requires each end user to know and disclose his/her discomfort function to a central entity (i.e. the aggregator). The MCA auction algorithm implements the VCG allocation and payments via an indirect market mechanism. In this way, end users are only required to respond to a specific sequence of queries, instead of being required to communicate their discomfort function. Thus, the mechanism does not require this direct-revelation and can also work with each participant solving an optimization problem in parallel, while still achieving the VCG outcome (and its nice properties). This allows for a privacy-aware implementation of an efficient and truthful B2C flexibility market architecture. In this subsection, we demonstrate how exactly the proposed (optimal and incentive compatible) mechanism can be configured with a scalable and privacy-preserving communication protocol. For this purpose, we exploit this related Kademlia work[18], although now that the mechanism is disengaged from the direct-revelation and central computation of the typical VCG, different communication protocols can also be applied.

The proposed mechanism is implemented through the development of a distributed communication protocol that preserves privacy (by doing all the necessary calculations required by MCA in a distributed fashion) while simultaneously guaranteeing all MCA objectives.

According to 1, the major requirements from distributed systems (with use cases in privacy preserving through blockchain) are: i) scalability in terms of number of users that participate in them (which according to 1 is directly related to the balanced distribution of the load/calculation overhead among participating users) and ii) low delay in terms of the time that is needed for the privacy-aware algorithm's execution (i.e. MCA's execution).

The proposed B2C flexibility market architecture exploits blockchain services 1, which are based on Distributed Hash Tables (DHT) 3 technologies, in order to execute MCA in a distributed fashion. In this context, end users do not inform the aggregator about their answers to the MCA's queries. Instead, these answers are sent only to a small subset of end users in N, in order to execute the calculations of MCA in a distributed fashion.

Thus, the proposed architecture acts as a substrate that offers a privacy preserving service to MCA through which participating end users cooperate in order to protect their personal data (i.e., their discomfort functions $d_i(\cdot)$) from the aggregator. In order to achieve this, they use a DHT 3, which is based on the scheme proposed by Kademlia 4. In Kademlia, each user (node) is identified by a number (nodeID) that can be seen as a point in a specific virtual space. The nodeIDs do not serve only as identification, but they are also used by the Kademlia algorithm to store and locate/get values/data hashes (i.e., the answers to the FSP queries). This process is realized through a peer-to-peer routing service (implemented in the network application layer) that Kademlia offers. In order to achieve this, each piece of information is given as input to a hash function, whose output belongs to the aforementioned virtual space. Each node is responsible for a subset of this virtual space according to its nodeID. Furthermore,

[18] P. Maymounkov, D. Mazières, "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric" in: Druschel P., Kaashoek F., Rowstron A. (eds) Peer-to-Peer Systems. IPTPS 2002. Lecture Notes in Computer Science, vol 2429, 2002.

participating nodes create and dynamically maintain routing tables in a bottom-up organized way. Thus, they can collectively reach any point of this virtual space, by exploiting their routing tables, in order to store and get information from this distributed data base. Through the exploitation of these functionalities, the distributed execution of MCA takes place through the use of 4 in the three following steps:

1. *Data insertion*: At each iteration k of MCA, each node i stores its bid $\widetilde{q_i^t}(\lambda_k)$ in another random node w through the use of the aforementioned DHT system. It is highlighted that w is different for each i and k (as it is derived from the output of the hash function that Kademlia uses), and in this way collusion of a relatively small number of users to acquire data, will fail (which is a requirement that 5 sets).

2. *Calculation of* $\zeta_i^k(\lambda_k)$: Kademlia organizes the participating nodes in a tree structure. The proposed system exploits this structure in order to calculate the sum $\sum_{i\in\mathcal{N}} \widetilde{q_i^t}(\lambda^k)$ that MCA requires. To do so in a distributed way, node j waits until all nodes with lower nodeID from it inform j on possible data values they have to send to j. This process continues recursively until the node with the highest id acquires the desirable data and then it calculates the sum. At this point, this node also requests and receives $D^t(\lambda^k)$ from the aggregator and checks the termination condition. If it doesn't hold, j proceeds by broadcasting $\sum_{i\in\mathcal{N}} \widetilde{q_i^t}(\lambda^k)$ and $D^t(\lambda^k)$ to all nodes, using the Kademlia tree. Thus, each node j calculates $\zeta_i^k(\lambda^k)$ by subtracting the $\widetilde{q_i^t}(\lambda^k)$ value that is stored in it (which is not its own $\widetilde{q_j^t}(\lambda^k)$ value, and it doesn't know whose it is).

3. *Final allocation and payments calculation*: at the next iteration $k+1$, a different instance of Kademlia tree could be created, so that $\zeta_i^{k+1}(\lambda^{k+1})$ is stored at a new node g, other than j. Thus, even in the case that a node is malicious, data privacy is not compromised. The tuple $A_i = \left\{ \sum_{\xi=1}^{k} \zeta_i^{\xi}(\lambda^{\xi}),\ \sum_{\xi=1}^{k} \left[\zeta_i^{\xi}(\lambda^{\xi}) \cdot \lambda^{\xi}\right] \right\}$, containing the allocation and payments of user i up until iteration k, is passed from user j to g. At the final iteration, the tuples A_i are communicated to the aggregator. Note that the aggregator receives only the final allocation and payments for each end user, i.e., only the sum of $\zeta_i^k(\lambda^k)$ and not the intermediate values $\zeta_i^{\xi}(\lambda^{\xi})$. This means that the aggregator (and any other node for that matter) does not have the data to estimate the entire local discomfort function $d_i(\cdot)$ of end user i.

The analysis above assumes that the aggregator is honest-but-curious. By this, we mean that the aggregator is curious to know the discomfort functions of end users, but is also honest and will never attack the system in order to acquire them. In case of a malicious aggregator (i.e. with no hesitations to break the law), more strict privacy assumptions are needed, but this case is outside of FLEXGRID's scope. The interested reader can refer to the recent literature of privacy-preserving aggregation for the smart grid 6 7 8 9 10 11.

4.5 Simulation setup and performance evaluation results

In this section, we present two detailed appliance models taken from the literature and then use simulations to demonstrate the advantages of the MCA and verify its properties. We also compare MCA with state-of-the-art approaches. Specifically, we compare it with the marginal cost pricing method **Error! Reference source not found.** in terms of truthfulness and

aggregator's profits and with the direct-revelation VCG method 12 in terms of scalability. Simulations were run in Matlab R2018b.

4.5.1 Detailed electric appliance models

The first model is taken from **Error! Reference source not found.** and includes appliances that control the temperature of an environment, such as HVAC units. The end user's most preferable temperature is denoted by parameter $T_i^{pref}(t)$ and was taken in our experiments to be uniformly distributed in the interval [75F, 79F]. The actual room temperature, denoted by $T_i^{in}(t)$, evolves according to:

$$T_i^{in}(t) = T_i^{in}(t-1) + \eta \cdot \left[T^{out}(t) - T_i^{in}(t-1)\right] + \theta \cdot \left(p_{i,HVAC}^t - q_{i,HVAC}^t\right), \qquad (4.12)$$

where $p_{i,HVAC}^t$ is the end user's measurable power consumption before the occurrence of a FlexRequest and $q_{i,HVAC}^t$ is the curtailment resulting from the FlexRequest. Clearly, we have:

$$p_{i,HVAC}^t - q_{i,HVAC}^t \geq 0, \qquad (4.13)$$

and we also have the operational constraint:

$$p_{i,HVAC}^t \leq \hat{p}_{i,HVAC}^t. \qquad (4.14)$$

In the experiments, $\hat{p}_{i,EV}^t$ was set to 5 kW. We considered end users living in the same geographical location, hence the outdoors temperature $T^{out}(t)$ for the whole day was the same for all end users and represented a typical summer day in Athens, Greece. Parameters η and θ where set to 0.9 and 3, respectively. The user's discomfort at timeslot t for such end users (appliances) was defined as the square difference between actual and desired temperatures:

$$d_i^{temp}(q_i^t) = \omega_{i,HVAC}^{temp}\left(T_i^{in}(t) - T_i^{pref}(t)\right)^2, \qquad (4.15)$$

where parameter $\omega_{i,HVAC}^{temp}$ expresses the end user's inelasticity in timeslot t and was randomly selected in the range [0.10, 0.50].

The second appliance model represents temporally flexible loads (e.g., EVs) and is taken from 14. The EV is plugged-in at timeslot γ_i, where γ_i is uniformly selected in the interval $[3,9]$, for one third of the end users and in the interval $[14, 20]$ for the remaining two thirds. Each EV charges with a charging power $p_{i,EV}^t$ and has a total charging demand of $E_{i,EV}$ kWhs, where $E_{i,EV}$ was uniformly selected in the interval $[6, 36]$. The end user wants the EV to be charged as soon as possible and any delay would bring discomfort. This model accurately represents en-route charging (where an end user stops to charge the EV in a charging station on his/her way to his/her destination). The desired charging duration, denoted as δ_i, was set to $\delta_i = 3$

timeslots for all end users. The upper limit on charging power, denoted by $\hat{p}_{i,EV}^{t}$, was selected as $\hat{p}_{i,EV}^{t} = \frac{E_{i,EV}}{\delta_i}$. That is, if FlexRequests occurred, each end user would charge his/her EV in 3 (consequent) timeslots. An EV operational constraint is given as:

$$p_{i,EV}^{t} \leq \hat{p}_{i,EV}^{t}. \qquad (4.16)$$

The EV cannot be charged before arrival:

$$p_{i,EV}^{t} = 0, \quad t < \gamma_i, \qquad (4.17)$$

and must be fully charged before leaving:

$$\sum_{t \in \mathcal{T}} p_{i,EV}^{t} \geq E_{i,EV}. \qquad (4.18)$$

During a FlexRequest, an end user may choose to curtail $q_{i,EV}^{t}$ units from EV charging and compensate for them by charging the EV in a later timeslot. This delayed charging (for timeslots after $\gamma_i + \delta_i - 1$), comes with a discomfort defined as:

$$d_{i,EV}^{wait}(q_{i\,EV}^{t}) = \sum_{t \in \{\mathcal{T} | t \geq \gamma_i + \delta_i\}} \left[\left(\omega_{i,EV}^{wait}\right)^{t-\gamma_i-\delta_i+1} \cdot p_{i,EV}^{t} \right], \qquad (4.19)$$

where parameter $\omega_{i,EV}^{wait}$ expresses the end user's inelasticity for delayed charging and was uniformly selected in the range $[1, 1.5]$.

4.5.2 Performance evaluation results

Over a time horizon of 24 timeslots, with a duration of 15 minutes for each timeslot and for a setting of 50 end users, we simulated a FlexRequest in timeslot 17 (where there was a peak in the aggregated energy consumption). The parameters of the reward function were set to $a = 3$ and $b = 0.05$. We used step $\varepsilon = 10^{-3}$ in the MCA algorithm (cf. Table above). The figure below (i.e. left hand side) depicts the aggregated consumption along all 24 timeslots. As the figure shows, there is a consumption curtailment in the timeslot where the DR-event takes place and a consequent shift of consumption to a later timeslot (timeslot 20). Note that it could not be shifted to timeslots 18 or 19, because constraints (4.14) and (4.16) were already tight for these timeslots.

Next, we investigate the effect that cheating has on the aggregator's profits, denoted by $\Pi^{truthful}$ for the case where end users act truthfully and by Π^{cheat} for the case where they act according to what brings them the highest utility. We plot the ratio $\Pi^{cheat}/\Pi^{truthful}$ for different values of end users' elasticities $\left\{\omega_{i,HVAC}^{temp}, \omega_{i,EV}^{wait}\right\}$.

To do so, for each experiment, we multiply the end users' elasticity parameters by a positive factor ω_f. Higher values of ω_f indicate more inelastic end users. The right hand side of the figure below shows that the ratio $\Pi^{cheat}/\Pi^{truthful}$ is maximized and is equal to 1 for the

MCA, verifying our theoretical results. We also observe that for the marginal cost pricing method 12, the aggregator's profit loss due to untruthfulness rises with ω_f (i.e., when end users are less elastic), indicating that our scheme's truthfulness property becomes more important in markets where participants are not particularly flexible.

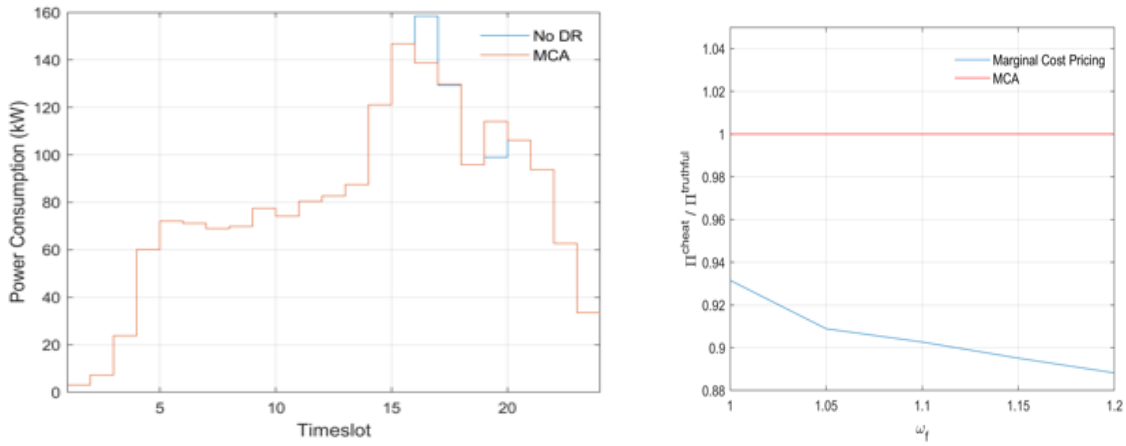

Figure 12: (a) Aggregated consumption as a function of time with and without the FlexRequest. (b) Ratio $\frac{\Pi^{cheat}}{\Pi^{truthful}}$ as a function of ω_f

Next, we verify *Corollary 1.* We simulated the FlexRequest for different values of the step parameter ε, measuring the proportional welfare loss: $W_{loss} = \frac{W_{opt} - W_{MCA}}{W_{opt}}$, where W_{opt} is the optimal welfare and W_{MCA} is the welfare achieved by the MCA. The simulation results in the figure below verify Corollary 1, which states that for small values of ε, the upper bound on the welfare loss grows linearly with ε.

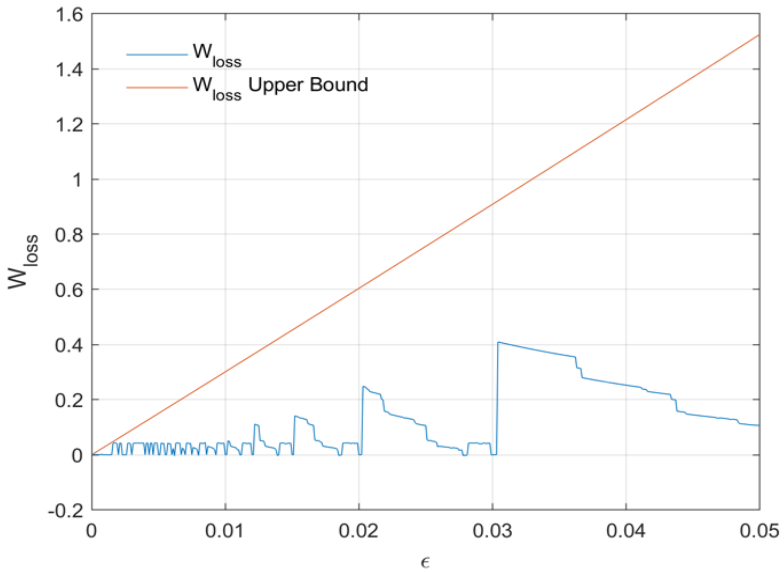

Figure 13: Proportional welfare loss of MCA as a function of the price step ε

As a next step, we compare MCA to the direct-revelation VCG method (proposed in 14), in terms of scalability with respect to the number of end users. Simulations have been carried

out on an Intel Core i7 4GHz, 64-bit, 16GB RAM, computer. As depicted in the figure below, the computational time of the method in 14 rises very quickly, which makes it impractical for real-time applications. In contrast, MCA scales remarkably well to any number of end users, since the algorithm's convergence time does not depend on the size of set $\mathcal{N}$. In order to confirm the scalability properties of our privacy preserving protocol presented in the previous section, we depict in the right hand side of the figure below the latency introduced by it. As it is known theoretically, in DHT technologies, this latency increases logarithmically with the number of end users. This is verified in the same figure, which testifies their outstanding scalability properties. In comparison to the timeslot duration (15 minutes, which is a typical granularity for measurements and clearing of the existing balancing markets in Europe), these results show that the proposed system is both scalable and efficient.

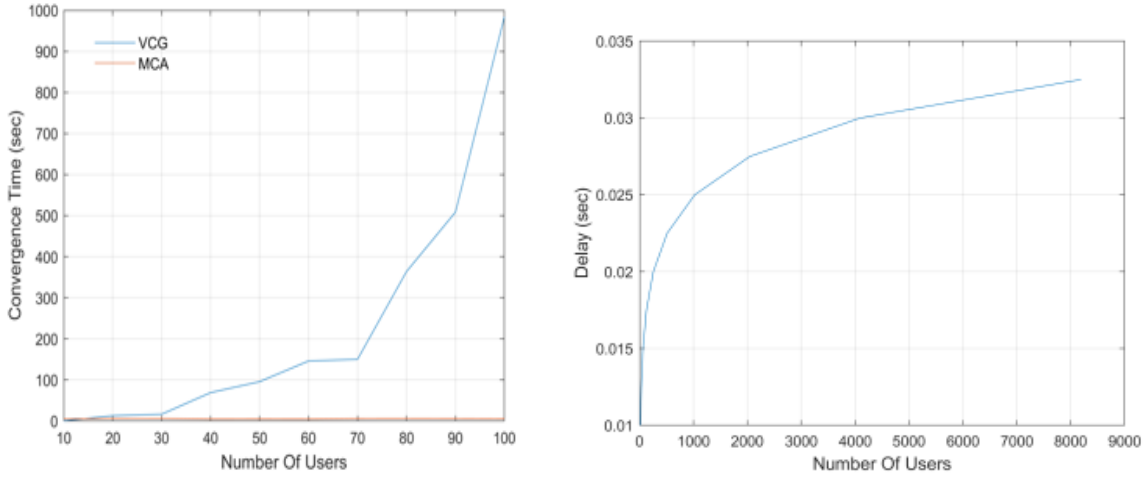

Figure 14: (a) Convergence time of MCA and VCG, as a function of the number of users. (b) Delay (latency) of privacy preserving protocol as a function of the number of participating users

Finally, we discuss the property of incentive compatibility. We thus verify the theoretical result of *Proposition 1* for end user models that satisfy the *Assumptions* and also experimentally study incentive compatibility in the case of inelastic appliances (where our *Assumptions* are not satisfied). Our results are compared against the typical Marginal Cost Pricing Method 12. We will now verify the truthfulness property via simulations. For the case of elastic end users, we assume that one end user misinterprets his/her discomfort by manipulating his/her $\omega_{i,EV}^{wait}$, while all other end users act truthfully. The untruthful end user is indexed by ch (for cheater). The cheater's utility U_{ch} is maximized for a certain choice of ω_{ch}. The figure below shows U_{ch} as a function of ω_{ch}.

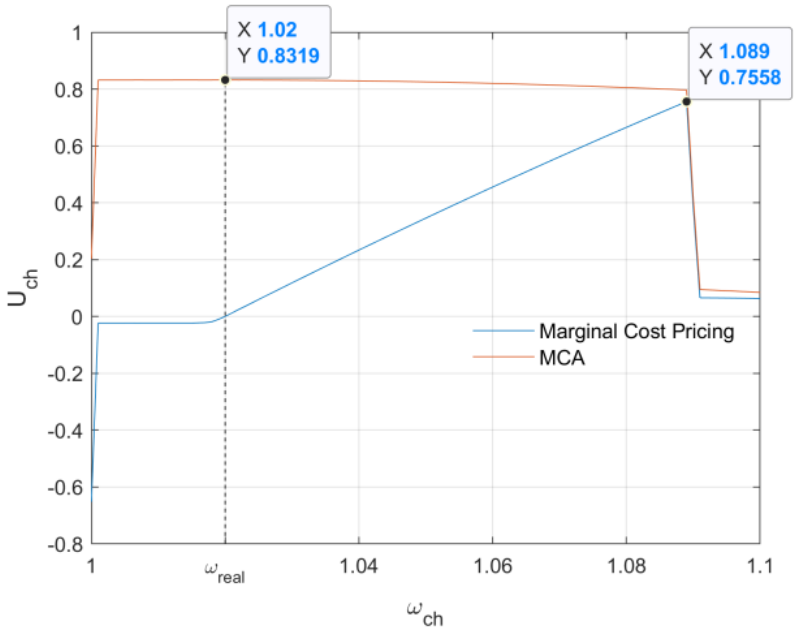

Figure 15: Focal end user's utility as a function of his/her choice of ω_{ch}

The black dotted line represents the focal user's actual (real) $\omega_{i,EV}^{wait}$, denoted as ω_{real}. For the MCA, the end user's optimal choice of ω (the one where U_{ch} is maximized) coincides with ω_{real}, which means that the end user's best strategy is to act truthfully. A similar statement cannot be made for the marginal cost pricing method.

The result of Figure 15 was expected, since it was already proven in Proposition 1. Although we cannot state a similarly strong theoretical guarantee for inelastic end users, nevertheless our simulations show similar results. We study the case where an appliance is inelastic, in the sense that it can only be turned on or off, but its consumption cannot take intermediate values:

$$p_{i,inel}^{t} \in \{0, \hat{p}_{i,inel}^{t}\}, \qquad (4.20)$$

and thus,

$$q_{i,inel}^{t} \in \{0, \hat{p}_{i,inel}^{t}\}. \qquad (4.21)$$

The end user's discomfort for turning his/her load off, is denoted by $d_{i,inel}^{off}$. Thus, the end user's discomfort function takes the form:

$$d_{i,inel}(q_{i,inel}^{t}) = \begin{cases} 0, & for\ q_{i,inel}^{t} = 0 \\ d_{i,inel}^{off}, & for\ q_{i,inel}^{t} > 0 \end{cases}. \qquad (4.22)$$

This kind of appliances violate $Assumption\ 2$. In fact, in mechanism design terms, the form of the end user's valuation for these appliances exhibits *complementarity* (the end user can either curtail all $\hat{p}_{i,inel}^{t}$ KWhs, but he/she cannot make use of an allocation that is smaller than $\hat{p}_{i,inel}^{t}$). It has been proven that in the presence of such complementarities that there is no tractable iterative auction that can achieve incentive compatibility 15. The clinching auction for example, could end up allocating any reduction between 0 and $\hat{p}_{i,inel}^{t}$ to the end user. In what follows, we present an extension of the MCA algorithm that accommodates inelastic end users and evaluate it via simulations. Although we can no longer theoretically guarantee

the property of incentive compatibility, nevertheless simulation results show that, in practice, truthful bidding is still the best choice for each end user.

Let I denote the set of inelastic end users. The first step is to run the MCA algorithm as described in Table 6. Then, we grant the MCA allocations $\zeta_i^k, \forall k$ only to elastic end users $i \notin I$. The remaining reduction $\sum_{k=1}^{\kappa} \sum_{i \in I} \zeta_i^k$, will be reallocated amongst the inelastic end users, in a way that respects constraints (4.21). This is actually an instance of the knapsack problem. In order not to compromise the computational time guarantees of our real-time auction, we use a simple heuristic to solve it. Inelastic end users are sorted in increasing order of their so called "bang-for-buck" i.e. their $\frac{d_{i,inel}^{off}}{\hat{p}_{i,inel}^t}$. Finally, we allocate $q_{i,inel}^t = \hat{p}_{i,inel}^t$ to user $i \in I$, in increasing order of the sorted list, until $\sum_{i \in I} q_{i,inel}^t \geq \sum_{k=1}^{\kappa} \sum_{i \in I} \zeta_i^k$. The procedure is depicted in Table 7.

Table 7: The Extended MCA algorithm

1.	**RUN THE MCA ALGORITHM (Table 6)**
2.	**SET** $q_{i,inel}^t = 0, \forall i \in I$
3.	**SORT USERS** $i \in I$, in increasing order of $d_{i,inel}^{off} / \hat{p}_{i,inel}^t$
4.	Set $q_{i,inel}^t = \hat{p}_{i,inel}^t$ for user $i \in I$, in increasing order of the sorted list, until $\sum_{i \in I} q_{i,inel}^t \geq \sum_{k=1}^{\xi} \sum_{i \in I} \zeta_i^k$

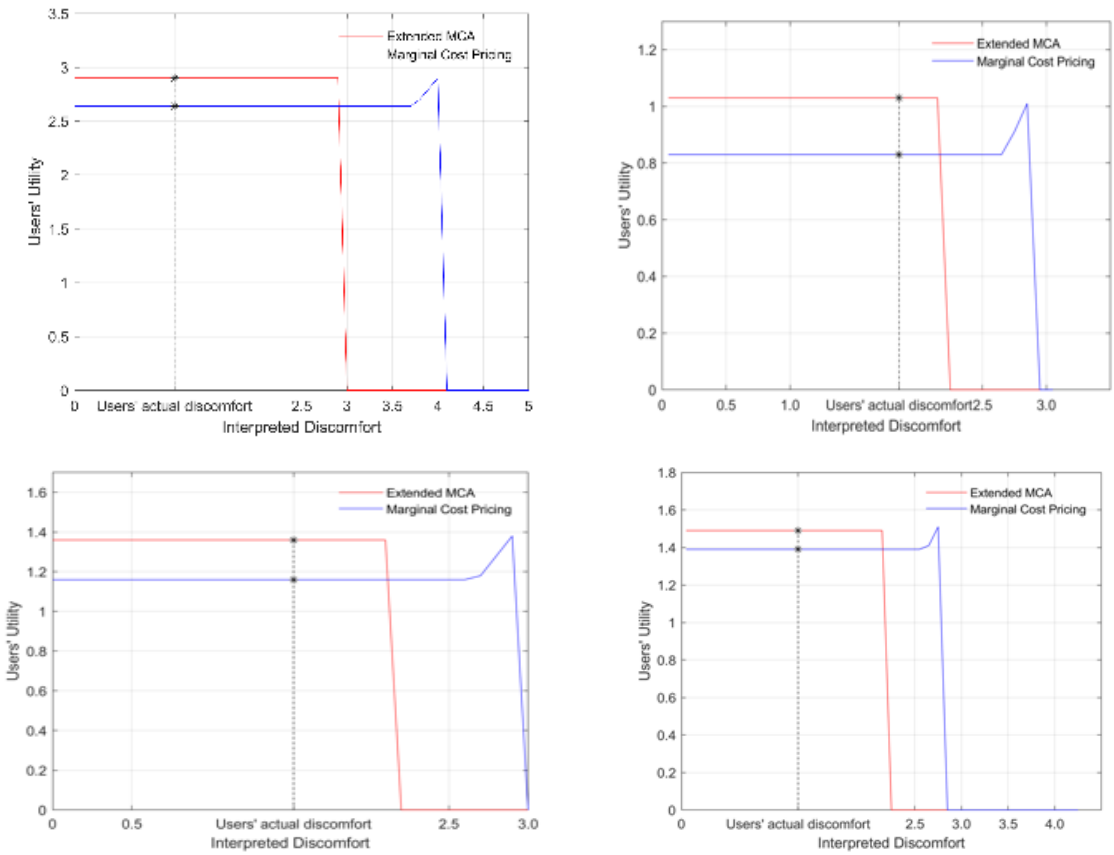

Figure 16: Users' Utility as a function of user's interpreted valuation

In Figure 16, we present indicative results (for various values of $\hat{p}_{i,inel}^{t}$ and $d_{i,inel}^{off}$), regarding the truthfulness of an inelastic end user's strategy, when participating in the extended MCA. More specifically, we tested how well an end user does (in terms of utility U_i, see Eq. 4.1), by interpreting his/her discomfort with various (untruthful) values. The end user's actual discomfort for curtailing $\hat{p}_{i,inel}^{t}$ units is marked with a vertical dotted line. From the figure, it becomes clear that the end user already achieves his/her maximum possible utility, by truthfully interpreting his/her discomfort and has nothing to gain by playing untruthfully. This is, again, in contrast to the marginal cost pricing approach 12.

4.6 Next research steps for the M19-M26 period

Within M19-M26, we will elaborate on the UCS 4.2 work in order to deal in more depth with algorithmic complexity and scalability problem. Our research findings indicate the need to deal with the scalability problem, which becomes very difficult to solve when we consider many 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[19]), more complex (and thus realistic) FlexAsset models and more stringent real-time constraints imposed by the B2C flexibility market.

In order to cope with these research challenges, our ongoing work focuses on combining the existing work on B2C flexibility market with an optimal cloud resource allocation algorithm. The cloud resource allocation algorithm will be 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 cost-efficient and competitive B2C flexibility market as a service.

Another research task, which is also related with respective WP6 work is to integrate the proposed Behavioral Real Time Pricing (B-RTP) scheme into the Automated Flexibility Aggregation Toolkit (AFAT) and FLEXGRID ATP. Thus, the aggregator user will be able to exhaustively run offline "what-if" simulations to decide about personalized FlexContracts that best fit with each end energy prosumer's needs and energy prosumption profile.

[19] Note that each end user may have several flexible electric appliances (FlexAssets), so the number of participating entities increases even more.

5 S/W integration in AFAT and FLEXGRID ATP

5.1 Summary of AFAT's functionalities and S/W development

The Automated Flexibility Aggregation Toolkit (AFAT) has been designed in a way that can be commercially exploitable as a standalone S/W toolkit, which can be integrated as a S/W "plug-in" in other larger S/W platforms developed by an energy aggregator company in the future. Within the FLEXGRID's context, AFAT will be integrated in the FLEXGRID S/W platform (ATP) and its operation will be tested via extensive lab experimentations and pilot tests within WP6 and WP7.

So far, in FLEXGRID, we have done the following work with respect to the AFAT:

- We have developed a first version of the AFAT's functionalities. In other words, we have developed and tested the first version of the research algorithms that will be running at the AFAT's backend. The initial research results are extensively analyzed and demonstrated in chapters 2-4 of the current document.
- AFAT's data modelling work has been finalized and is provided in D6.1[20] (M18). In particular, for each one of the three main algorithms to be integrated in AFAT, we have designed the APIs for the interconnection between the: i) AFAT's backend services, ii) AFAT's frontend services, and iii) central FLEXGRID database.
- As part of WP8 business modeling work, we have identified the AFAT's Key Exploitable Results (KERs) and have already made a qualitative business analysis regarding the ways that the proposed functionalities can be further exploited in the commercial aggregator's business. More details are provided in D8.2 (M18).

From M19 onwards, we will continue the WP3 research and will start integrating the first version of the algorithmic solutions in the AFAT. Then, we will extensively test and validate our algorithms in FLEXGRID ATP at TRL 5. In the figure below, the progress of AFAT's development is depicted throughout the whole project's lifetime as follows:

- Within WP3, we conduct high-quality scientific research work by developing advanced mathematical models and algorithms beyond state-of-the-art and publish them in high-quality scientific journals and conferences (TRL 3).
- After the extensive testing and validation of the proposed algorithms at TRL 3, the next step is the deployment of REST API servers and REST API client for the integration of AFAT's frontend and backend services.
- The next step is the testing and validation of the AFAT algorithms via the use of FLEXGRID ATP at TRL 5 (WP6).
- Finally, within WP7 work, we will conduct small-scale real-life pilot tests of AFAT's functionalities in the UCY pilot (TRL 6).

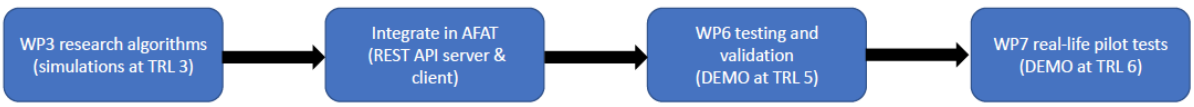

Figure 17: Progress of AFAT's development and respective technology readiness levels (TRLs)

[20] https://flexgrid-project.eu/deliverables.html

5.2 AFAT's frontend services

The aggregator user will be able to use a bunch of services from the FLEXGRID ATP. Once logged in the ATP via a single sign-in authentication process, the aggregator user will be re-directed to the AFAT's frontend services. The Graphical User Intefaces (GUIs) will be based on the existing WISECOOP application, which has been developed with H2020 WISEGRID project[21]. The goal of FLEXGRID is to use WISECOOP as a S/W substrate based on which the FLEXGRID's WP3 algorithms will be integrated.

AFAT's frontend (GUI) will be comprised of three basic tabs, namely:

- Manage a FlexRequest
- Create a FlexOffer
- Manage a B2C flexibility market

By using the "Manage a FlexRequest" tab, the aggregator user will be able to visualize the profit of accepting a FlexRequest and the "consequences"/remaining flexibility of its portfolio after the positive response. The goal is to deviate from the baseline only by the amount of energy in FlexRequests, which were accepted by the aggregator. Two modes of operation will be considered as follows:

- Online operation: A new FlexRequest-Dispatch is published in real-time by a FlexBuyer in the ATP. The aggregator is instantly informed and then will run the UCS 4.1 algorithm (cf. chapter 2) to decide the updated dispatch per FlexAsset / end user that belongs to its portfolio.
- Offline operation: 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 algorithm will run iteratively.

As of the "Create a FlexOffer" tab, the aggregator user will be able to visualize a FlexOffer and then submit (post) it in FLEXGRID ATP at a specific time instance regarding its participation in the DLFM market. Then, the FMO user will also be able to visualize this FlexOffer as well as the DSO (i.e. FlexBuyer). If this FlexOffer is not accepted in DLFM, it may be forwarded to the TSO's balancing market. Two modes of operation will be considered as follows:

- Online operation is when the aggregator user wants to create 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 operation is when the aggregator user wants to run "what-if" scenarios to see whether it is more beneficial to participate in the existing TN-level balancing market or DN-level balancing market (i.e. DLFM).

Finally, regarding the "Manage a B2C flexibility market" tab, the aggregator user will be able to visualize in AFAT frontend (i.e. ATP) several KPIs that make him/her recommend a new

[21] See more technical details about WISECOOP application here: https://www.wisegrid.eu/project-tools

(more beneficial) FlexContract to a set of end energy prosumers. Only offline operation is considered as follows :

- Offline operation: The aggregator user 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. Only the aggregator user will be able to visualize the results.

5.3 AFAT's backend services and integration in FLEXGRID ATP

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

- A flexibility aggregation algorithm that tries to find the optimal scheduling solution of available FlexAssets in order to respond to a FlexRequest. The proposed solution is based on a centralized optimization model and is described in detail in chapter 2 (cf. UCS 4.1).
- A flexibility aggregation algorithm that automatically aggregates availability flexibility in price/quantity tuples for a given future timeframe. The proposed solution is extensively described in chapter 3 (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 is described in chapter 4 (cf. UCS 4.2).

For all the above-mentioned algorithms, a detailed data model (i.e. algorithmic inputs and outputs) is presented in chapter 5 of D6.1. For each one of the three algorithms, there will be a tab in the AFAT frontend. Once the aggregator user clicks on one tab, s/he will be able to configure/customize/fill in the input parameters that are needed for each algorithm to be able to run. Once the aggregator user clicks on the "Run algorithm" button, step 1 process that is shown in the figure below, will be followed. More specifically, the API client that resides at the AFAT frontend will automatically gather all input parameters and will send them to the API server that resides at the AFAT backend.

After the AFAT backend receives the input parameters, the next step is to request for the required input data from the FLEXGRID central database (DB). More specifically, an API client that resides at AFAT backend requests for input data from an API server residing at the central DB. In step 3, the input data is retrieved, and now the algorithm can be executed.

Once the algorithm produces the results, these output parameters will be automatically gathered by the AFAT-ATP API and will be sent to the AFAT frontend so that the aggregator user can visualize the results in a comprehensive and user-friendly manner. The final step (i.e. step 5) is for the aggregator user to understand the results and if s/he is interested in further elaborating them, then s/he can optionally select to store them in the central DB in order to be able to retrieve, visualize and possibly compare them with other results in the future.

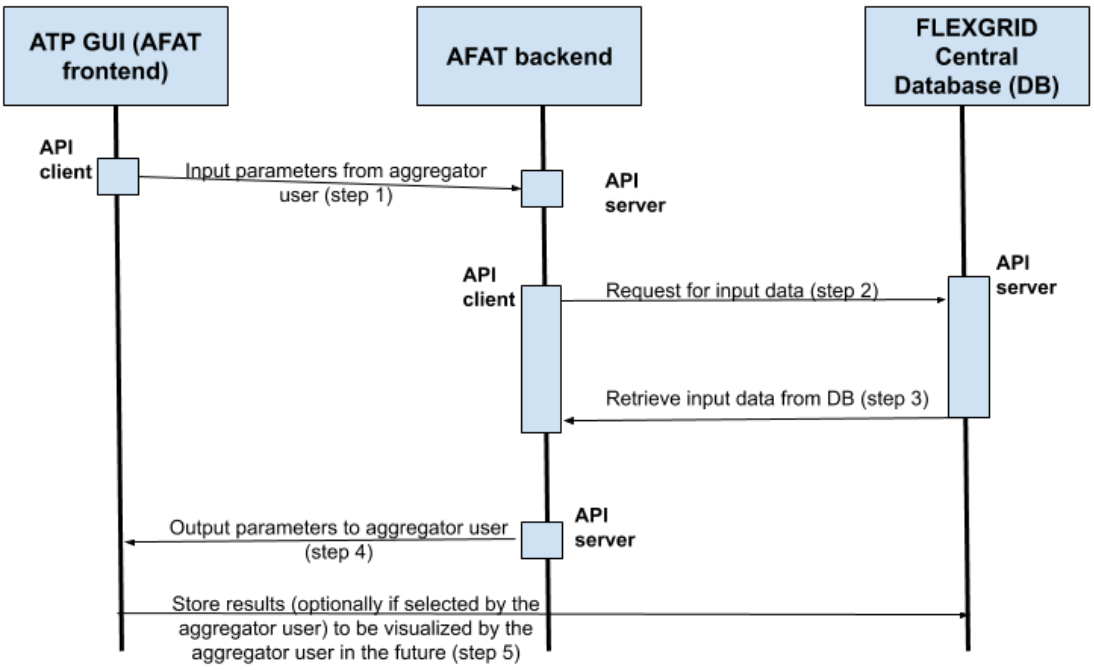

Figure 18: Sequence diagram for the S/W integration of WP3 research algorithms in AFAT and FLEXGRID ATP

6 Conclusions and next steps

In the following months, WP3 partners will progress the current research work presented in this report and will provide the final research results in Month 26.

Regarding UCS 4.1 work, UCY will follow the research plan described in section 2.6. As of UCS 4.2 and 4.3 work, ICCS will follow the research plan described in sections 4.6 and 3.6 respectively.

In the figure below, the timeline schedule of WP3 is illustrated. Milestone #5 has been achieved with this deliverable, while one more milestone remains to be accomplished for month #26 with the submission of D3.3.

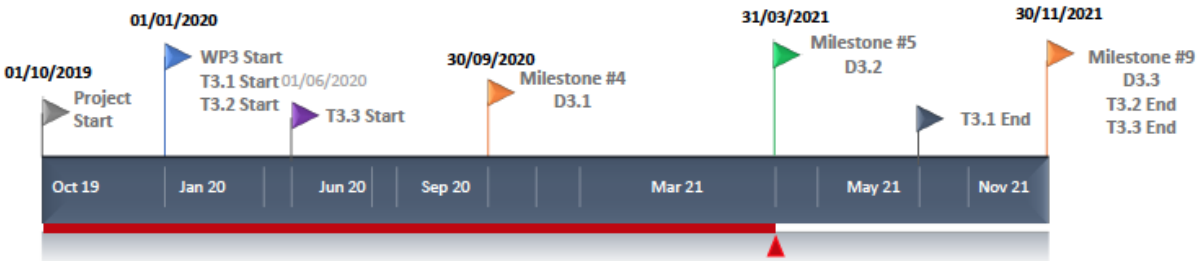

Figure 19: Current FLEXGRID project's WP3 timeline schedule (MS 5 has been accomplished)

References

1 Felber, P., Kropf, P., Schiller, E. and Serbu, S., 2014. “Survey on Load Balancing in Peer-to-Peer Distributed Hash Tables”, IEEE Communications Surveys and Tutorials, vol. 16 no.1, pp.473-492.

2 G. Zyskind, O. Nathan and A. Pentland, "Decentralizing Privacy: Using Blockchain to Protect Personal Data," 2015 IEEE Security and Privacy Workshops, San Jose, CA, 2015, pp. 180-184.

3 Xylomenos, G., Ververidis, C.N., Siris, V.A., Fotiou, N., Tsilopoulos, C., Vasilakos, X., Katsaros, K.V. and Polyzos, G.C., 2014. A survey of information-centric networking research. IEEE Communications Surveys and Tutorials, vol. 16 no. 2, pp.1024-1049.

4 P. Maymounkov, D. Mazières, "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric" in: Druschel P., Kaashoek F., Rowstron A. (eds) Peer-to-Peer Systems. IPTPS 2002. Lecture Notes in Computer Science, vol 2429, 2002.

5 Z. Baharlouei and M. Hashemi, "Efficiency-Fairness Trade-off in Privacy-Preserving Autonomous Demand Side Management," in IEEE Transactions on Smart Grid, vol. 5, no. 2, pp. 799-808, March 2014.

6 H. J. Jo, I. S. Kim and D. H. Lee, "Efficient and Privacy-Preserving Metering Protocols for Smart Grid Systems," in IEEE Transactions on Smart Grid, vol. 7, no. 3, pp. 1732-1742, May 2016.

7 F. Knirsch, G. Eibl and D. Engel, "Error-Resilient Masking Approaches for Privacy Preserving Data Aggregation," in IEEE Transactions on Smart Grid, vol. 9, no. 4, pp. 3351-3361, July 2018.

8 P. Gope and B. Sikdar, "Lightweight and Privacy-Friendly Spatial Data Aggregation for Secure Power Supply and Demand Management in Smart Grids," in IEEE Transactions on Information Forensics and Security, vol. 14, no. 6, pp. 1554-1566, June 2019.

9 D. Egarter, C. Prokop and W. Elmenreich, "Load hiding of household's power demand," 2014 IEEE International Conference on Smart Grid Communications (SmartGridComm), Venice, 2014, pp. 854-859.

10 A. Awad, P. Bazan and R. German, "Privacy Aware Demand Response and Smart Metering," 2015 IEEE 81st Vehicular Technology Conference (VTC Spring), Glasgow, 2015, pp. 1-5.

11 H. Li, X. Lin, H. Yang, X. Liang, R. Lu and X. Shen, "EPPDR: An Efficient Privacy-Preserving Demand Response Scheme with Adaptive Key Evolution in Smart Grid," in IEEE Transactions on Parallel and Distributed Systems, vol. 25, no. 8, pp. 2053-2064, Aug. 2014.

12 P. Samadi, A. H. Mohsenian-Rad, R. Schober, V. W. S. Wong and J. Jatskevich, "Optimal Real-Time Pricing Algorithm Based on Utility Maximization for Smart Grid," 2010 First IEEE International Conference on Smart Grid Communications, Gaithersburg, MD, 2010, pp. 415-420.

13 N. Yaagoubi and H. T. Mouftah, "User-Aware Game Theoretic Approach for Demand Management," in IEEE Transactions on Smart Grid, vol. 6, no. 2, pp. 716-725, March 2015.

14 P. Samadi, A. H. Mohsenian-Rad, R. Schober, V. W. S. Wong, "Advanced demand side management for the future smart grid using mechanism design", IEEE Trans. Smart Grid, vol. 3, no. 3, pp. 1170-1180, Sep. 2012.

15 F. Gul and E. Stacchetti. Walrasian equilibrium with gross substitutes. Journal of Economic Theory, vol 87 no 1, pp. 95–124, 1999.
16 T. Li, S. H. Low, and A. Wierman, “Real-time flexibility feedback for closed-loop aggregator and system operator coordination,” arXiv2006.13814, 2020.
17 Tin Kam Ho, “Random decision forests,” in Proceedings of 3rd International Conference on Document Analysis and Recognition, vol. 1, 1995, pp. 278–282 vol.1.