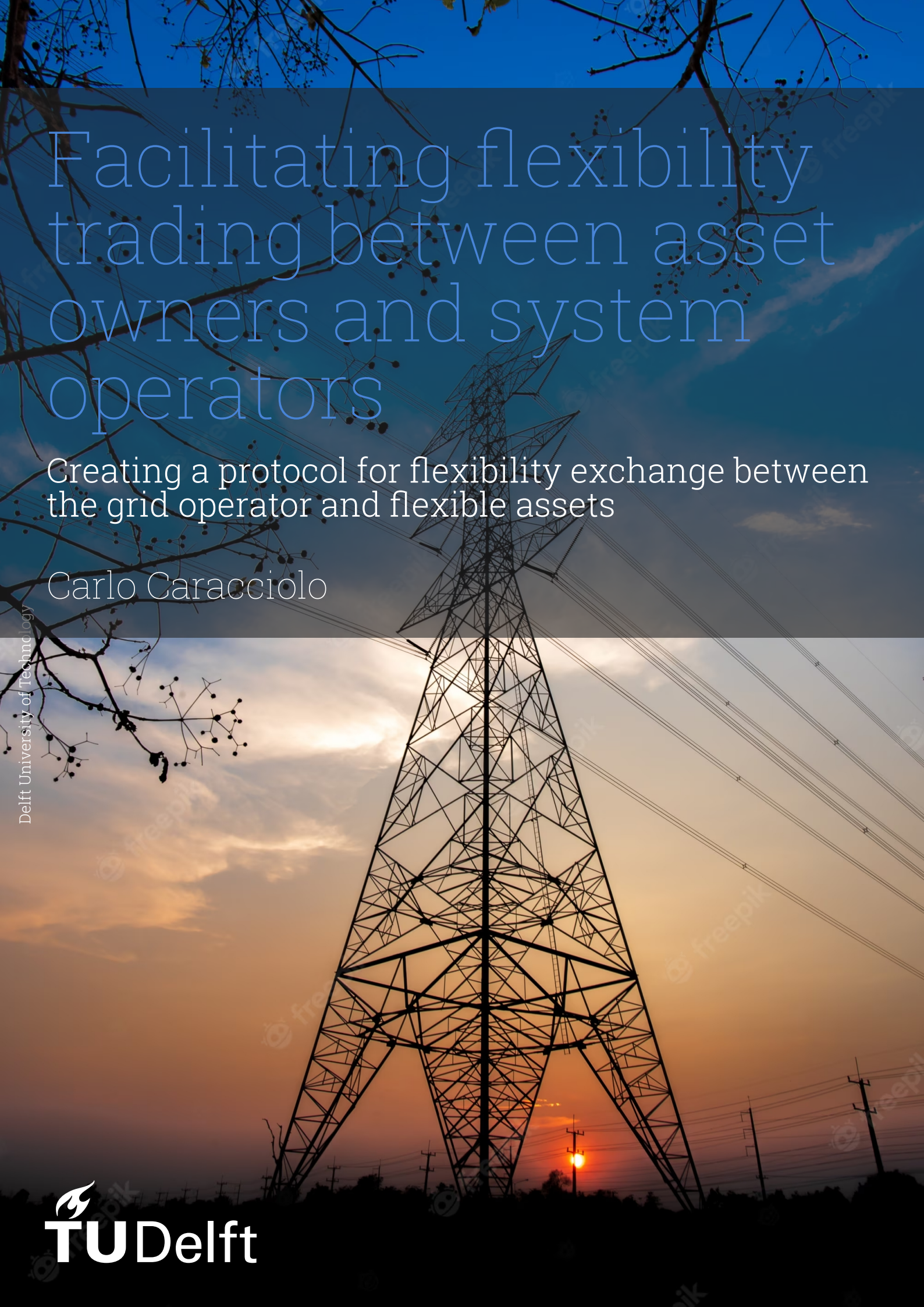


Facilitating flexibility trading between asset owners and system operators

Creating a protocol for flexibility exchange between the grid operator and flexible assets

Carlo Caracciolo

Delft University of Technology

TU Delft

Facilitating flexibility trading between asset owners and system operators

Creating a protocol for flexibility exchange between the grid operator and flexible assets

by

Carlo Caracciolo

Student Name	Student Number
Carlo Caracciolo	4465210

Thesis supervisor:	Peter Palensky
Daily supervisor:	Milos Cvetkovic
Project Duration:	December 2022 - June 2023
Faculty:	Faculty of Electrical Engineering, Mathematics and Computer Science, Delft

TU Delft

Acknowledgements

This thesis has been very interesting to me and I am glad that I was able to deepen my knowledge in the world of balancing and flexibility. With the current state of the grid and the necessity of a solution, I believe that flexibility will play a major role. What I await with eagerness is when flexibility at the residential level will be readily available and mainstream. The value unlocked both technically as well as commercially is immense and it really fascinates me. Eventually I would love to see a world where energy is locally produced and exchanged, creating the eutopian scenario of power by the people and for the people.
I would like to thank my supervisor Milos Cvetkovic for pointing me out on such an interesting subject and guiding me throughout the project. I truly enjoyed the brainstorming session that we had together with Aihui Fu, I felt like my contributions were taken into account which I really appreciated.
I would like to also thank my family and friends for giving me support not only during the thesis but also throughout my study and I am grateful for all the feedback that I got. It is through the multiple interaction that my passion for technology grew.
With this document, my master terminates and although I am ready to move on to the next step, I am very grateful for the journey.

Carlo Caracciolo
Delft, June 2023

Abstract

As the electrical grid becomes more constrained and grid reinforcement/expansion is no longer the only viable solution, electric flexibility is slowly becoming a more practical approach. Additionally, with the increasingly greater share of flexible devices being deployed, there is immense potential to solve this problem. While flexibility provision is already an implemented market mechanism, it mostly revolves around large industries that have more predictable behavior. At the residential or distribution level, said flexibility is harder to harvest due to the unpredictability and heterogeneity of the systems and actors involved.

This report's focus is creating a protocol facilitating the flexibility exchange between asset owners and the grid operator. In order to bridge these two actors, an aggregator program is created with the task of managing in a responsible way these exchanges. The protocol has used The Green Village, an aggregate of smart residential housing located on the TU Delft campus, as a physical system to base the protocol and program. Although The Green Village has been used as a reference, the protocol and program should be versatile for any application. The main goal, when developing the protocol, was to have the aggregator program take in as many tasks related to flexibility exchanges as possible to increase compatibility (interoperability). The protocol and aggregator program were also designed to facilitate modifications and upgrades (plug-and-play) while preventing communication errors (redundancy). To fulfill these requirements, the report takes the following structure.

First, the different methods and frameworks enabling flexibility as well as involved actors are discussed. Then the protocol and aggregator program are explained in depth. Finally, a validation through simulation is presented and inspected.

Contents

Nomenclature

Abbreviations

Abbreviation	Definition
AGR	Aggregator
TGV	The Green Village
EMS	Energy Management System
DSO	Distribution System Operator
DR	Demand Response
MPC	Model Predictive Control
DT	Digital Twin

1

Introduction

1.1. Context

In the last decade, the world has focused on transitioning towards a cleaner world. This change has led to a shift from appliances using gas to electricity resulting in an increase in energy demand from the electric grid. Additionally, the growth in population and businesses has contributed to supplementary stress on the grid. In cases such as the northern neighborhood of Buiksloterham-Zuid and Overhoeks in Amsterdam, large to medium consumers can no longer be accepted as the grid reached its maximum capacity with expansion plans only scheduled for 2023 [1].
To make things more complicated, more and more decentralized renewable generation is being installed at the distribution level. This ever-increasing uncontrollable energy source represents a large injection of energy in the grid when consumption is low, leading to further congestion management issues. This was the experience for the German grid and photovoltaic integration at the low-voltage grids [2].
To counter this, different solutions are available. Grid reinforcement/expansion is usually a solid solution as it increases the allowed capacity. For example, the southern German distribution system operators (DSO) have been upgrading their operating equipment mainly to avoid overloading due to the growth in PV systems installations [2].
However, such a measure requires time and financial/material resources that are not always available which is why it is usually perceived as a long-term solution. For example, in the previously mentioned situation of the neighborhoods of Buiksloterham-Zuid and Overhoeks in Amsterdam, the DSO Aliander has evaluated that the grid reinforcement/expansion would take 5-7 years. Additionally, the DSO has concluded that by the time the grid reinforcement would be completed, the demand would have already outgrown the reinforcement, ending up in the same situation as before [1]. Another solution available to solve grid congestion is called demand response. It is faster and cheaper to implement with respect to grid reinforcement and is usually perceived as a temporary solution. One type of demand response is called demand side flexibility (or flexibility for short) and is defined as the ability to adapt and change generation and consumption [3]. Flexibility can then be used to make sure that the network capacity is not surpassed (congestion management) but also for providing energy in parts of the network that may need but not have enough available, although the former is the focus of this research.

1.2. Congestion Management with flexibility

Congestion management through flexibility can be done with two methods, namely the implicit and explicit methods. The implicit method is when the user changes their power consumption in function to price signals [3] [4]. For example, users will change their consumption more towards the middle of the day, when typically, there is low consumption and high production of RES leading to a lower energy cost. On the other hand, explicit flexibility is when flexibility is traded onto an energy market and the provider receives financial compensation for it [3] [4]. Demand response solutions are interesting because they can be regarded, to a certain extent, as part of a mandatory transformation happening to the electric grid. As previously explained, the capacity at the distribution level is continuously growing which disrupts the traditional electricity grid structure of one-directional centralized power generation to a bidirectional decentralized model. DSO's role is transitioning into a dynamic and essential piece of the puzzle. The DSO must be able to monitor and efficiently manage the energy flows to avoid setbacks and down time with demand response becoming an important featured responsibility [5].
Research has shown positive outcomes in using demand response flexibility for congestion management [3]. For example, a community battery coupled with an intelligent algorithm proved to be effective for the reduction in grid congestion [6]. Additionally, a test pilot with an energy community in Heerhugowaard (EnergieKoplopers) using the USEF framework has shown positive outcomes with the community preventing congestions both time-wise (congestion lasted a shorter time) as well as height (gravity of the congestion) [7].
Although the positive outcomes, demand side flexibility and flexibility markets at the distribution level still have a long journey ahead and need development both on the technical side as well as policies [3] [6]. Market-based flexibility, albeit maturing, is still defined as a niche practice [8] with still mismatches in the provided flexibility which makes it suboptimal and hard to rely on [7]. For example, the EnergieKoploper test pilot had one-third of its sold flexibility not served with the main root of the cause being bad planning and coordination between the flexible assets and actors involved [7].
There are different ways that an asset can offer flexibility, and one of them is through an aggregator. An aggregator (AGR) is responsible for coordinating the provision of flexibility to the grid. As the name suggests it aggregates loads that create its flexibility portfolio which, through its know-how, tries to maximize its revenue by selling the flexibility. The AGR thus represents the energy community in the flexibility market by bidding flexibility and/or trading it depending on the market mechanism used. In this study, the aggregator coordinates with an energy community made of residential loads coupled with local photovoltaic production and storage. To provide flexibility, the aggregator communicates with the Energy Management System (EMS) of the energy community, a central brain that plans and takeS decisions on how the community energy assets should operate. The EMS objective is to make sure that the user's comfort is not exposed and thus is keener on conserving energy to match the user's demand at all times.
There is a conflict of interest between the aggregator that tries to maximize its financial benefits by extracting flexibility and possibly exposing the user's comfort and the EMS that tries to do the opposite.

1.3. Research Question and report structure

The goal of this study is to create a protocol that can coordinate the negotiations between the AGR and the EMS. As the EMS is responsible for tasks that are not directly linked with flexibility, i.e. controlling the assets to ensure the correct functioning of the system, and does not have any intrinsical aim to provide flexibility, it makes the most sense to consider the AGR as the missing link for such coordination. The AGR, as defined previously, takes in the duty of managing the flexibility bids. More details on the diversity of the tasks will be described later on. Such a requirement partially defines the outline of the protocol. As a variety of EMS using different control methodologies may be involved, it is crucial that the protocol can be easily implementable. Together with the proposed protocol, comes a Python code, called AGR program. Said code computes flexibility from information given by the EMS and sells this flexibility to the grid operator (in this case DSO). It is important to add that the communication between

the AGR and the grid operator uses the USEF framework which shapes the way the EMS-AGR interaction is designed. Both the protocol and code are made open source in order to boost the usage and they are by no means considered perfect. In fact, one of the structural requirements of the code is to be flexible, meaning that parts of it should be interchangeable while staying operational. The purpose of such design is that any user that decides to implement the code and, if desired, can tailor it to their own preferences. Such change/addition should be errorless until the foundation of the messages between the different involved actors is not modified. Eventually, the resulting communication structure should ease and strengthen the incorporation of flexibility for energy communities through AGR while not undermining the users' preferences. Thus, the research question of this paper revolves around the following:

How to create a protocol that enables aggregators to trade flexibility in a responsible manner?

To do so, the following sub-questions will be treated, each meant to clarify and elaborate the different processes and actors that are involved in the flexibility exchange. The questions are as such;
First, what are existing methodologies that facilitate flexibility trading?
Second, who are the involved stakeholder and what are their interests?
Third, what are the developed trading protocols?
Fourth, how can the AGR program architecture be created to facilitate usage and modifications?

To answer each of these the body of the report is divided into six chapters. First, the importance of standardization for flexibility trading is explained. Secondly, the physical system and the actors involved in the flexibility trading process are described. Then, the trading protocol and more precisely the USEF Framework is examined. The fifth chapter goes into depth about the protocol working and the different sections of the AGR programs. Then the sixth chapter discusses the simulation results. Finally, a conclusion and future recommendations are given about the study.

2

The importance of standardization

This chapter explains the importance of standardization for the deployment of flexibility. First standardization in controlling flexible assets is discussed, more precisely the s2 protocol and OpenADR. Then some metrics to evaluate and operate flexibility are explained and compared. Finally, the framework for flexibility trading is briefly introduced.
At the end of this chapter, the reader will have an overview of the different technics that are involved in flexibility.

2.1. Standarization

As previously presented, the energy network has many challenges ahead and flexibility seems to be a compelling solution. With plenty of research and test pilots, it leads to the question of what limits the implementation of this technology.
During the development of the EnergieKoploper test pilot, extensive research on the flexible asset's owner was carried out to understand the need and preferences. An important aspect that had been determined during the interviews was that consumers were prone to participate until their comfort would not be affected. The flexibility needed to be achieved without any effort from the consumer, attainable only with trust in the organization and system. While different factors influence said aspects, standardization plays an important role in obtaining them. As explained in the Energiekoploper test pilot report, standardization will help to have reliable, uncomplicated, and uniform automatic control of smart devices. This will not only ease the implementation of flexibility in households and other systems but will also assist to lower investment barriers [7].
Another feature that has great importance is that such standards (and protocols) are also open source to incentives the use and thus deployment of flexibility. This is particularly important for residential applications as the impact on grid congestion is greater when more households are involved in the process. Such logic is valid at any level of the system too. For example, an open standard for an EMS can allow interoperability between smart devices of different brands enabling scaling up and reusability of results [9].

2.1.1. Controlling flexible assets

The control of flexible assets has been the focus of TKI Urban Energy and RVO, who published an investigation paper of standards for communication of smart devices and other assets with central control (i.e. EMS). As stated in the paper, there is no standardized interface available on the market that

enables the universal communication between an EMS and assets from different manufacturers each with their own protocols [9].
The paper then introduces their solution: the S2 protocol which is part of the EN 50491-12, a standard created by the European Commission that lays out the general requirements for Home and Building Electronic Systems and Building Automation and Control Systems.
However, before discussing how the S2 protocol works, it is important to understand the two approaches an EMS can communicate with a smart device to unlock flexibility.

The first one is the direct approach. As the name suggests, the EMS directly communicates with the smart device to modulate its consumption and deliver flexibility. To do so the EMS uses the device's protocol. The struggle with such an approach is that the EMS is likely to interact with various devices manufactured by different manufacturers that use different protocols. The EMS has then two options: use devices that operate on the same protocol, drastically constraining the range of flexibility possibilities, or build a polyglot EMS. A polyglot EMS has many protocols implemented and thus can interact with different devices. However, the problem with such EMS is that some of the protocols may be proprietary and the owner/company may not want to share it [9].
Figure 2.1 shows the schematic of two systems using the direct approach. One EMS uses only devices with the same protocol and the other is a polyglot EMS.

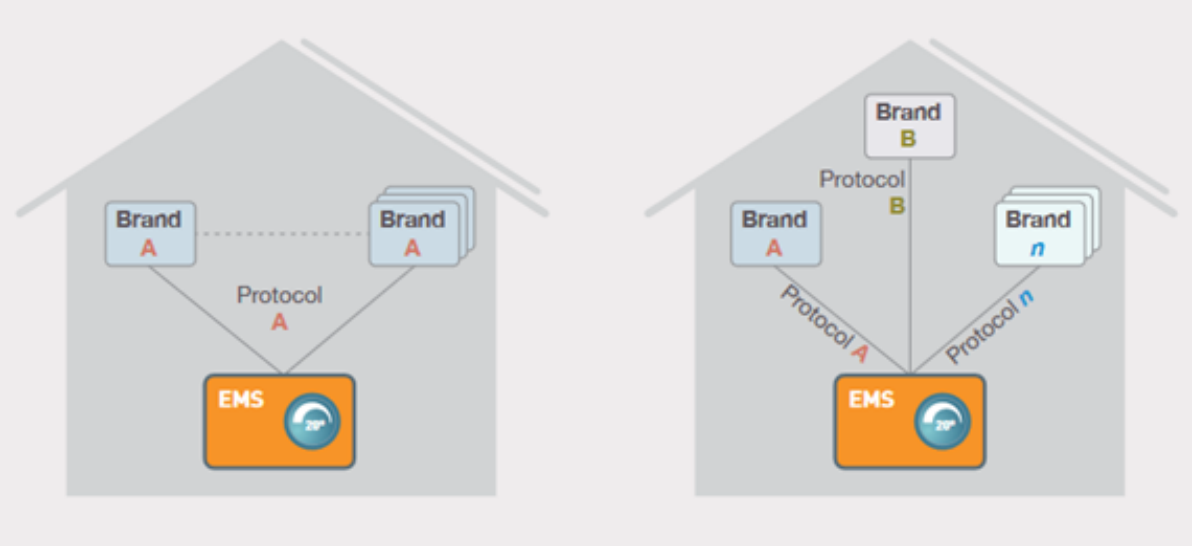


Figure 2.1: Schematic of direct approach: one protocol EMS (left) and "polyglot" EMS (right). [9]

The second way is the indirect approach. This approach circumvents the need for using only the same protocol or a polyglot EMS by introducing a middleman in the system. The EMS does not directly communicates with the device but rather with a resource manager (RM) that acts as a connection point between the EMS and the smart devices. The RM communicates with the smart device with their protocols and "translates" the messages to the EMS through a standardized protocol namely, the S2 protocol [9]. As a shortcoming, the position of the RM can affect the system performance (i.e. slower response time due to many layers) but also maintenance issues. This may be an issue as, in the EnegieKoploper test pilot, often flexibility could not be delivered because the system lagged behind, although it was reported that the impact is small [7]. Additionally, at the time of writing this paper, the S2 protocol has not yet been implemented and thus still lacks proof of concept [9].

Figure 2.2 shows the schematic of how the EMS and the flexible/smart devices are connected and communicate when the S2 protocol with the RM is implemented.

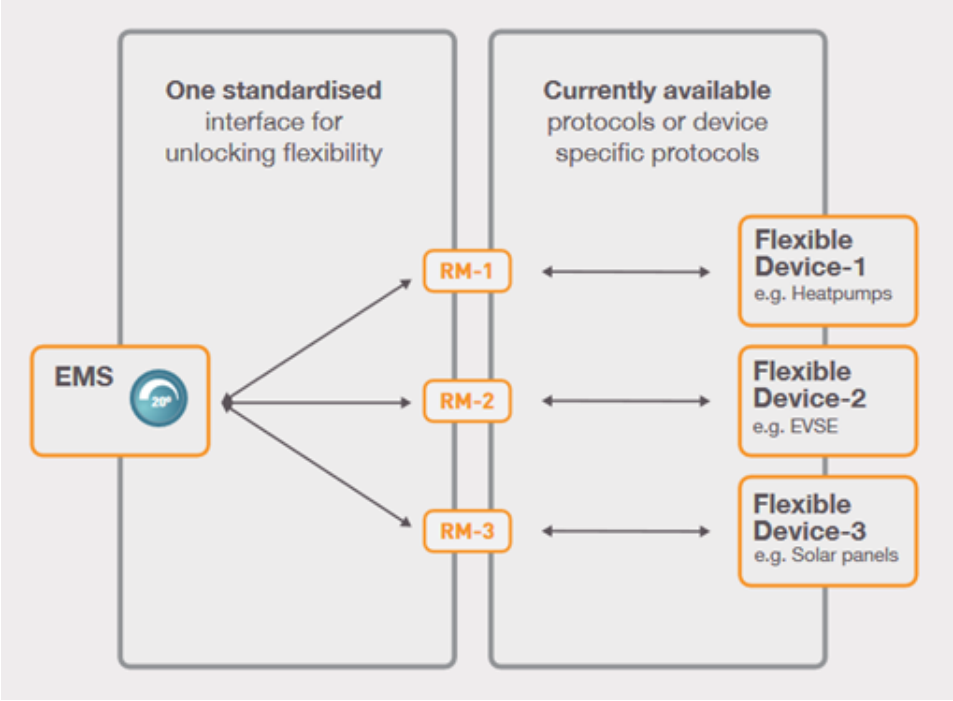


Figure 2.2: Schematic overview of EMS to device communication via a Ressource Manager [9].

The TKI Urban Energy and RVO document also underlines the importance that the indirect and direct approaches should not be considered as opposite but rather complement as the RM to smart device communicate through a direct protocol. [9]

There are other instances of standardized protocols for communication between system actors. Another example is the OpenADR which stands for Open Automated Demand Response.
Its goal is to facilitate communication between energy actors such as energy providers, aggregators, and end users by creating an open flexible data model for common information exchange that anyone can implement. This helps to have a reliable, repeatable, robust, cost-effective, and automated DR at the customer level [10].
The message transmission between the involved actors is done by assigning the roles of the virtual top node (VTN) and the virtual end node (VEN). The VTN is the sender of messages while the VEN is the receiver and the messages are transmitted over a secured internet connection. An example of a VTN and VEN communication would be a system operator sending in a flexibility request to an aggregator that in his turn will send a message to an EMS to activate the flexibility. However, just because one actor is a VTN does not mean that it cannot become a VEN in another interaction. In the case described above, the AGR is a VEN when receiving information from the system operator but in the next step, it becomes the VTN when communicating with the EMS. Figure 2.3 shows the possible relationship between VTN and VEN.
OpenADR offers a variety of different services and functions used for requesting and activating demand response. Such services and functions vary based on which profile is used [10]. Nonetheless, such profiles can be further modified by including more services. An example is the OpenADR and USEF Foundation partnership where three new DR program templates were added to better fit the framework's purpose [11]. More detail on the USEF framework operation is given in chapter 4.

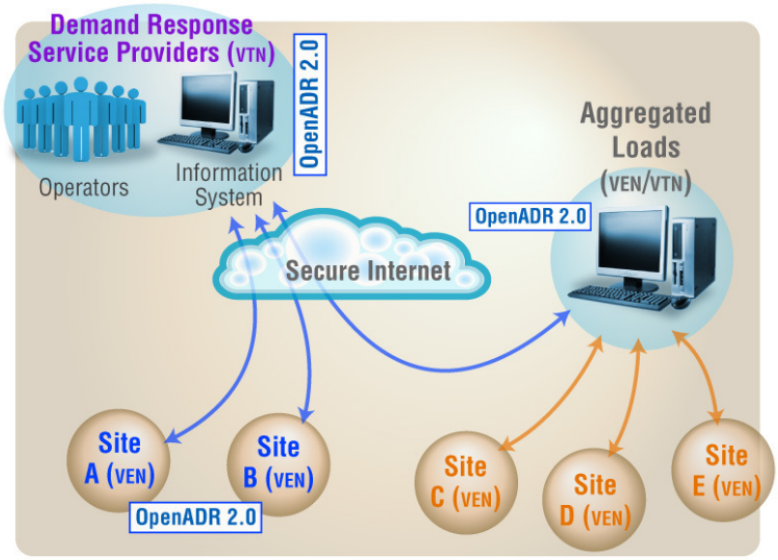


Figure 2.3: Possible relationship of VTN and VEN [10].

2.1.2. Metrics to evaluate and operate flexibility

Standardization research has also been done on how to forecast and evaluate flexibility. If done incorrectly, the consequences can be harsh penalties [12] but could eventually also result in grid failure if the unserved flexibility volume is large and the system operator is unable to fix the failure in time [4]. During the EnergieKoploper test pilot, one-third of the flexibility was not delivered due to (in part) bad forecasting of load as well as available flexibility [7]. One way that the authors of the report think that it can be avoided is through highly available information. Metrics have thus been developed with the intent to facilitate the computation and decision-making of flexibility. This subsection discusses two metrics and then compares them.

The first paper, *Quantification of operational flexibility from a portfolio of flexible energy resources*, uses the simulation of different scenarios to evaluate how a system is expected to serve its flexibility. Using the metrics proposed by the author, the operator can analyze how the system performs. If it redeems that too many scenarios have an unacceptable outcome, then it can ease the boundaries of the system, run again the simulations, and see if the result is acceptable. The metrics used to assist the operators are the following: the Expected Unserved Flexibility Energy (EUFE), the Expected Duration of Insufficient Flexibility (EDIF), and the Expected Flexibility Index (EFI).

The EUFE calculates the average expected unserved flexibility over all scenarios. It is expressed as Watt hours (Wh) and its value is how much the operator should expect its system to not be able to serve the needed flexibility.

The EDIF is a graphical overlook over the time horizon of the simulation that indicates at which PTUs the system will not serve its flexibility. It uses a heatmap to distinguish the volume and direction of unserved flexibility (i.e. yellow is used for a large volume of unserved flexibility). Such graphical overlooks can help the operator to differentiate the volume and PTUs when the system does not serve flexibility according to the simulation. The operator can use it to extrapolate patterns and change the boundaries accordingly to see if it helps to reduce the unserved flexibility.

The EFI is a unitless metric that measures the operational flexibility of the system. The metric gives the percentage of how much of the expected flexibility will be served. The closer the number is to one, the more flexibility is expected to be served. For example, an EFI equal to 0.9 means that 90% of the total planned flexibility is expected to be served. As the paper explains, with the outcome of the EUFE and EFI combined with the graphical overlook of the EDIF, the operator can have a clear understanding of which DR action plan and policies are more interesting. The metrics evaluate the flexibility of a portfolio

of assets as well as the consequence of an operational policy [12].

The second paper, *Integrating Energy Flexibility in Production Planning and Control - An Energy Flexibility Data Model-Based Approach*, introduces the Energy Flexibility Data Model (EFDM).
The EFDM is structured such that it can give clear insights to operators wishing to activate flexibility. As the paper explains, the EFDM has the intent of "generically describing flexibilities and flexibility measures using key figures and technical parameters". In simpler words, the EDFM is used to describe the flexibility possibilities that a system can undertake by comparing the reference operation and the possible variation in performance. To do so four classes are used:
The Flexible load, which describes the assets or the interaction between assets that can lead to a change in energetic performance (i.e. reduction of energy consumption).
The Dependency, correlated to how assets are interdependent from each other. The usage or ramping up of one asset may affect the operation of another (i.e. the charging of a battery means that it cannot be used to power another asset).
The Storage, to characterize which assets or interactions can be used to store energy. This refers to batteries, heat but also inherent energy storage (the latter are more common in industrial settings).
Flexibility load measure, is used to give the concrete power changes within the flexibility space that the system can withstand by using the 3 classes explained above [13].

While the two paper's metrics may have the same purpose, to give a clear insight into the system's flexibility to operators to ease their decision-making, there are some differences that need to be pointed out. As explained, the EDFM is used to define the flexibility possibilities. The system knows what its possibilities and limitations are and then, therefore, plans how the flex can be used. The EUFE, EDIF, and EFI, on the other hand, focus more on quantifying the lack of flexibility. The metrics are used to calculate the amount of flexibility that will not be served. From this, the operator can determine how much it wants to reduce this lack of flexibility by altering the boundaries of how much flexibility the assets can deliver [12][13].

2.1.3. Frameworks for trading flexibility

Last but not least, the USEF Framework needs to be discussed. It has been already explained that the framework is employed to coordinate the exchange of flexibility. Through USEF, the grid operator (i.e. DSO) is able to send a request for flexibility at a specific node. All registered actors (i.e. an aggregate of prosumers), connected at the node can bid to serve said flexibility, and a local flexibility market.
The USEF Foundation has done extensive research work to create a standardized framework that contains a protocol of communication between the grid operator and the participants. Just as discussed in the previous examples of standardizations, the USEF Framework has been created to ease and fasten the deployment of flexibility. [4]

This chapter has highlighted the importance of standardization in different fields. First, it has been explained that standardization can facilitate the implementation by helping to have reliable, uncomplicated, and uniform automatic control of smart devices. This also reduces the financial barriers which also incentives the installation and usage. The second important aspect of standardization that has been discussed is that it enables interoperability which can allow scaling up and reuse of previous data and/or system structures. As an example, the S2 protocol was presented. Through an indirect approach, it is able to connect and control flexible assets that operate on different protocols, although with some limitations nonetheless. The importance of open source was also discussed, with the example of OpenADR which facilitates the information exchange between energy actors. Thirdly, various metrics to forecast, plan and evaluate flexibility were discussed. The EUFE, EFI, and EDIF, help to evaluate how much flexibility a system can provide. The EFDM, helps to understand how activating flexibility affects the reference operation through the guidance of four classes. While the two sets of metrics have similar goals, it was explained how the way they tackle the situation is different. Finally, the standardization of trading protocols was mentioned with the USEF Framework briefly discussed.

3

Physical system and actors involved

This chapter treats on the physical system used as well as the different actors involved. First, an explanation is given of the installation's characteristics that this study is based on. Then the different actors involved in the flexibility trading are introduced as well as their pros and cons with respect to flexibility trading. Finally, the structure of the whole system is presented and discussed.

3.1. Physical system

The system used in this research is The Green Village (TGV) developed by TU Delft. Its purpose is to be a field lab for sustainable innovation and has been modeled with the intent to be an off-grid neighborhood. The system architecture is composed of centralized self-generation and storage. The energy generation is done thanks to a solar panels installation with a peak power of 6 kW. While the storage has a 14 kWh battery as well as 27 cylinders storing 60 kg of H2 under a 300 bar pressure. The system is also equipped with a 2.3 kW fuel cell that allows the conversion of hydrogen into electricity but also enables the re-usage of heat. The neighborhood is also equipped with flexible devices such as heat pumps [14].
The centralized system is operated by an EMS that plans and controls the energy consumption of the community. The EMS uses a Model Predictive Control (MPC) that forecasts and plans the behavior of the system over a time horizon, in function of controllable variables [15].

3.2. Actors involved

3.2.1. The Distribution System Operator

Depending on what framework is used for flexibility trading, the involvement of the DSO may differ. As this study focused on the USEF Framework, it will be taken as the reference model. More explanation of the Framework is given in chapter 4 but in short, the DSO is responsible for forecasting congestion, requesting it, and managing the flexibility market (i.e. clearing the market) [4].

3.2.2. The Energy Management System (EMS)

The EMS is a centralized commanding unit that automates the monitoring, control, and optimization of the system that it controls to best fulfill the goal that it has been assigned [16].
With respect to flexibility trading, the EMS has the advantage of having a diverse portfolio. For example, TGV is equipped with a storage unit, a self-generation asset (i.e. photovoltaic on the rooftop), and flexible consuming assets like a heat pump. These multitudes of assets give the EMS the possibility to adapt and change its consumption and generation in a multitude of ways. This means that the EMS has many different ways that it can supply flexibility to the market.
The other advantage of the EMS is that it directly controls the flexible assets. No third party needs to be involved to activate the flexibility of the assets as the EMS is the one in charge of operating them.
However, the EMS presents some disadvantages with respect to flexibility trading. The EMS is created not only to automate the monitoring of a system for more efficient control but to do all while remaining within the asset owner's comfort boundaries. The EMS "works" for the asset owner and will try to not expose the user's comfort rather than risking it in trading flexibility thus endorsing more of an energy-conservative mindset. It is important to note that this also depends on what goal and boundaries were given to the EMS. Furthermore, the EMS manufacturer may not want to implement flexibility trading in their algorithm for three main reasons.
The first one is latency. Having to include flexibility computation can not only lead to a slower EMS, which is quite unwanted but might also be out of reach for certain manufacturers. For example, price forecasting is needed to set a price on the volume of flexibility power bids. This is an expertise that certain companies fully specialize in (i.e. N-Side's energy market forecast [17]) and may be out of scope (or budget) for an EMS manufacturer.
The second reason is impact. A single EMS may control a lot of different flexible assets but only a small capacity. While it may have many different options to provide flexibility, the impact that one single EMS may not be enough to make a difference and thus the EMS manufacturer may not have any or enough financial incentive to implement trading. Additionally, the market entry criteria creates an entry barrier for EMS controlling relatively small capacities further disincentivizing the manufacturers [18]. One of the entry barriers is a minimum resource bid size, varying for each country. In the case of the Netherlands, a 4 MW resource capacity is needed which hinders the market entry for residential, commercial, and even small industrial consumers which leaves them excluded from the balancing market [18]. Depending on the market structure, it can be requested that an actor must be able to do symmetrical bidding. This means that the actor must be able to provide flexibility by both reducing or injecting energy into the grid. This further excludes flexible asset owners that may be equipped with an asset of only one kind (i.e. only PV or only heat pump) [18].
Finally, the third reason is responsibility. As explained previously (and in more detail in chapter 4), flexibility that has been sold but not served results in a penalty (fine). This can push away EMS manufacturers who may not want to take in liabilities for unserved flexibility.

3.2.3. The Aggregator (AGR)

The AGR role is defined by the European Commission as the combination of multiple customer loads and/or electricity generation for sale, purchase, or auction in an electricity market [19]. There are many different types of AGR that have varying roles. For example, the integrated AGR takes care of supplying energy to its clients while as well providing flexibility to the grid [20]. For the case study of this paper, a contractual AGR is considered, meaning that it is not responsible for providing energy to the clients but only for coordinating and serving flexibility [20].
The AGR presents advantages with respect to flexibility trading. The first one is that it is an entity that is fully focused on profit-maximizing and has the know-how to participate in the electricity market [18]. If the market is designed properly and the AGR receives the correct signal, through its portfolio, it can provide effective flexibility which results in increased economic well-being and efficiency for both the parties involved (i.e. asset owners) and the power system [20]. This is usually true as market prices indicate the network status and thus if the assets are coordinated correctly then it will yield an improved

system condition [20]. The AGR also takes all liabilities for unserved flexibility, leaving the asset owners free of trouble [21].
On the other hand, there are also some disadvantages. If the AGR is not equipped with a diverse portfolio, its flexibility possibilities are reduced and it will not be able to deliver efficiently. Thus there is a need to have enough capacity to be eligible for the market but also that said capacity is diverse enough to be able to efficiently participate [21]. Additionally, because of its economically driven mindset, flexibility assets owner's interest may be neglected, leading to a possible high financial yield but at the cost of reduced user comfort. This is a condition that can lead to flexible owners not wanting to put at disposition their assets [21].

3.2.4. Conflict of interest but complementing features and system architecture

From the previous subsection, we have seen how the EMS and AGR can both participate in the balancing market. On one side there is the EMS, which directly controls flexible assets with the intent of maximizing the owner's comfort which leads to being less likely to trade flexibility (conservative approach) as well as excluded by the regulations (minimum capacity to participate) and driven away by the practical reasons (EMS manufacture has no incentive to include flexibility trading algorithms).
On the other side, there is the AGR which knows how to trade flexibility and can take liabilities for unserved flexibility but because of its economically driven mindset, its flexibility bids may not reflect the flexible asset owner's preferences.
This leads to a conflict of interest between the AGR which wants to maximize the financial yield and the EMS which wants to maximize the user's comfort. Thus a compromise must be found between the two which is why the architecture is shown in figure 3.1.

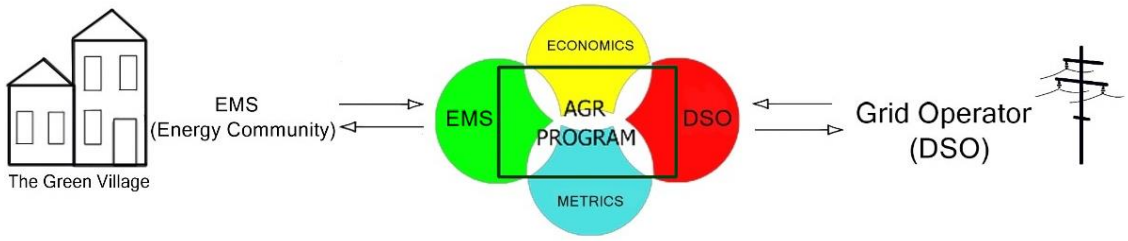


Figure 3.1: System architecture for flexibility trading

The AGR is placed between the EMS and the DSO and trades flexibility on behalf of the EMS. It bridges the interaction between the EMS and the DSO. However, the AGR does not have full control of the flexibility bids but rather it takes care of the following task:

- It is in charge of evaluating the flexibility possibilities and creating flexibility bids that can be submitted to the system operator.
- It is legally responsible for serving flexibility. This means that it is liable for any unserved flexibility and not the EMS.
- It is responsible to communicate with the EMS and DSO to try to best match each other's requests.

The EMS is also assigned some tasks:

- It directly controls the flexible assets. Makes sure to have them deliver flexibility.
- It administrates the flexibility bidding to make sure that the flexibility bids do not neglect the user comfort.

Such system architecture makes sure that each actor performs the tasks that it is most skillful at. On one side the AGR is concerned with all the computation and legal requirements to provide flexibility and on the other, the EMS ensures that the flexibility is served and that its provision does not influence the user's comfort.
It is important to note that this architecture is not new and aggregation at the residential level is already happening although large customers (i.e. industrial loads) are the more common practice [20].

This chapter presented the different components that form the system. the Green Village (TGV) was introduced together with the flexible assets that it has installed and the control algorithm that operates the assets. Thanks to the MPC EMS, the TGV can plan its consumption and operation Additionally, a diverse and large storage system, allows it to function off-grid.
The different actors involved in the flexibility trading were also discussed. First, the DSO, although briefly mentioned, is responsible for forecasting congestion but also managing the flexibility market. Then the EMS, which controls the flexible assets and, in the case of TGV, has a diverse portfolio. However, because of a variety of reasons, the EMS is discouraged to take part in flexibility trading. The AGR on the other hand, does not directly controls the assets but trade flexibility. With a know-how experience, it can trade flexibility if it has a diverse portfolio. However, its economically driven mindset can lead to undermining users' preferences and deter EMS's willingness to participate.
The AGR and EMS have a conflict of interest but also complementing features. This duality can be advantageous if each actor is assigned the correct role. In the system architecture discussed, the AGR is placed in between the EMS and the DSO, bridging the interaction between the two. The AGR is responsible to evaluate the available flexibility as well as taking all liability for serving it. The EMS operates the assets so that the flexibility can be provided but also makes sure that the AGR does not overstress the system and undermine the asset owner's preferences/comfort.

4

The Trading Framework

This chapter discusses the trading framework that already exists. The GOPACS and USEF Framework are discussed, with the latter being explained in more detail. A brief comparison between the two is given. Then, the PNNL's Transactive Energy Framework is explained.

4.0.1. USEF Framework and GOPACS

It has been explained that flexibility markets and mechanisms already exist and are active. An example of a market is GOPACS. Although it is not a market platform on its own, it uses other market platforms to allow grid operators to coordinate and solve congestions by trading flexibility. When a congestion happens at a node, the responsible grid operator will send in a sell order which will be met by a buy order from another grid operator in a non-congested area and thus resolve the congestion [8]. Before the sell and buy offers are confirmed an algorithm check whether the buy and sell orders do not create another congestion somewhere else in the network [22] [8].
An example of a mechanism and market is the previously mentioned USEF framework. Such framework creates a communication protocol between the DSO and actors to standardize the exchange of flexibility between the two. By allowing the communication between the two parties, the DSO can publish requests of flexibility and the concerned parties can bid their willingness to provide a chosen volume of said flexibility (either all of it or just a part) and thus creating a flexibility market [4].
To do so the framework describes 5 consecutive phases that map the communication between the AGR and DSO and the actions that need to be taken.

The first step is the Contract Phase. This phase determines how the AGR will participate in the flexibility market. It is at this stage for example that bilateral contracts are arranged. This is usually done once and does not need to be repeated every day. Signing a bilateral contract usually gives the DSO the possibility to request flexibility during a period. Figure 4.1 shows the UFTP information exchange in the Contract phase.

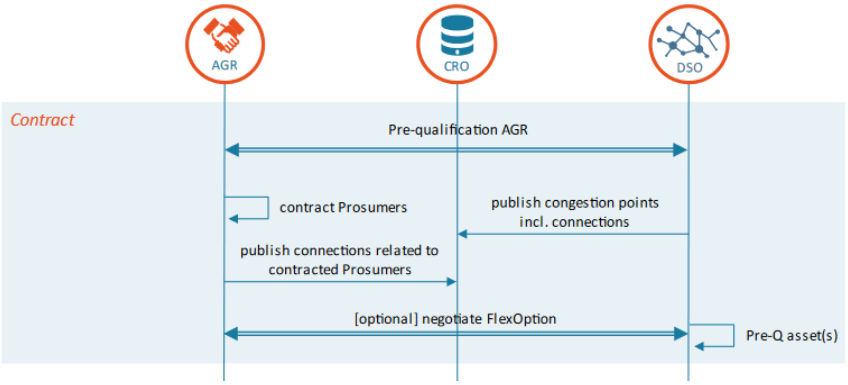


Figure 4.1: UFTP information flow in Contract phase [4]

The second step is the Plan Phase. During this phase all actors wishing to participate in the flexibility market will have to register at the network node they are connected to. The AGR shares a list of which nodes its assets are connected and equivalently the DSO shares the lists of nodes where flexibility can be provided. The outcome of this second phase is that the DSO knows the 'active' flexibility participant at each node and the AGR knows which of its assets are connected at a node that could possibly require flexibility. The flexibility market is therefore set up and all the active actors at each node are defined. This process needs to be done daily as the assets of an AGR may change, it is also at this stage that the DSO reserves a volume of flexibility that it may want to employ in case a bilateral contract has been made with an AGR. Figure 4.2 shows the UFTP information exchange in the Plan phase.

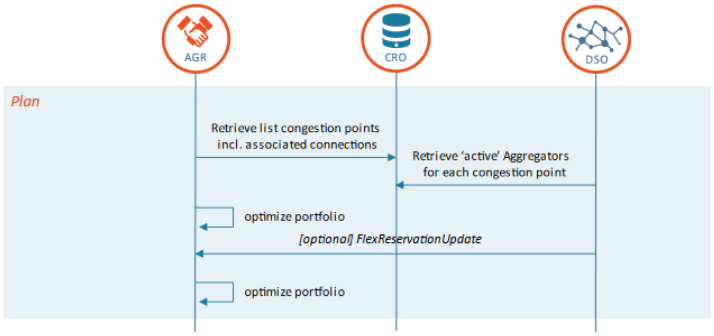


Figure 4.2: UFTP information flow in Plan phase

The third step is the Validate Phase. AGRs need to provide a D-prognosis or baseload for the day ahead (next full calendar day). The D-prognosis is the load curve (both consumption and production as one) of each imbalance settlement period (ISP). The DSO will use all the D-prognosis as well as information about the power flow to understand whether there will be any foreseeable congestions at that node. If a (or multiple) congestion is apparent, the DSO will send in a request for flexibility known as FlexRequest. All actors registered and connected to the node can then send an offer with the amount of flexibility that they are willing to offer. This is called FlexOffer. A FlexOffer can be either a reduction of power or an injection of power in the grid. The former can be done either by consuming at a different ISP when the power consumption is not as large and the grid is not congested (this is known by the AGR during the Plan Phase) or by using storage. It is up to the DSO to decide which offer it prefers. The DSO will send in a FlexOrder which will confirm to the AGR that its offer has been accepted. The AGR then must send a new D-prognosis with the flexibility incorporated. The FlexOffers will normally be determined through merit order but the DSO still has the last word and can decide however it wants. For day-ahead trading, the USEF closure time is 2 hours before the gate closure of the national balancing regime, while for the intra-day trading, the closure is 1h before the first ISP for which the D-prognosis has been calculated and also includes the D-prognosis of all the remaining ISP of that calendar day [23]. Figure 4.4 shows the two market closures for both the day-ahead and intraday. Figure 4.1 shows the UFTP information

exchange in the Validate phase.

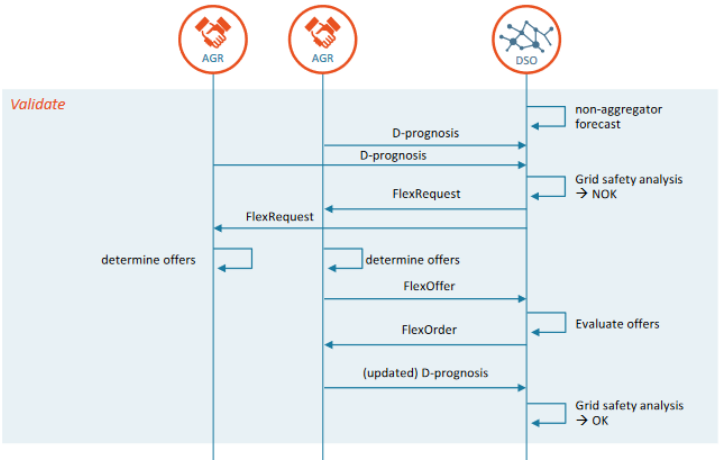
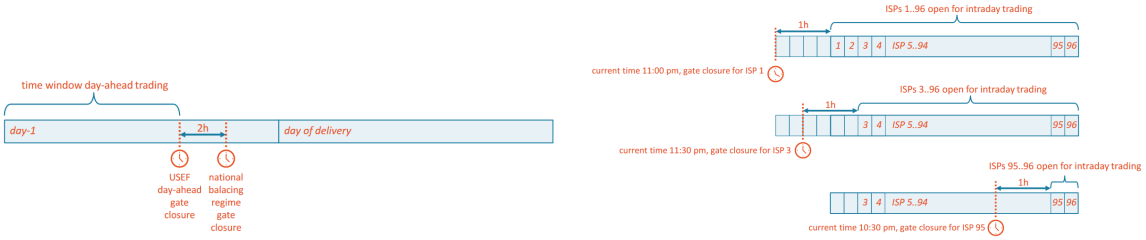


Figure 4.3: UFTP information flow in Validate phase [**USEF_validate**]



(a) USEF day-ahead gate closure time [23]

(b) USEF intra-day gate closure [23]

Figure 4.4: The two different USEF market gate closures.

The fourth step is the Operating phase. The day ahead has finally arrived and every time instance has been planned and validated. Nonetheless, variations in the baseline (or D-prognosis) at the node are likely to happen and adjustments will be needed. The steps of the Validate Phase are repeated except within a shorter timeframe as now the FlexOffers and FlexOrders will be determined 1h before the first ISP to which the D-prognosis applies. This can be regarded as an intraday flexibility market.
Figure 4.5 shows the UFTP information exchange in the Operate phase.

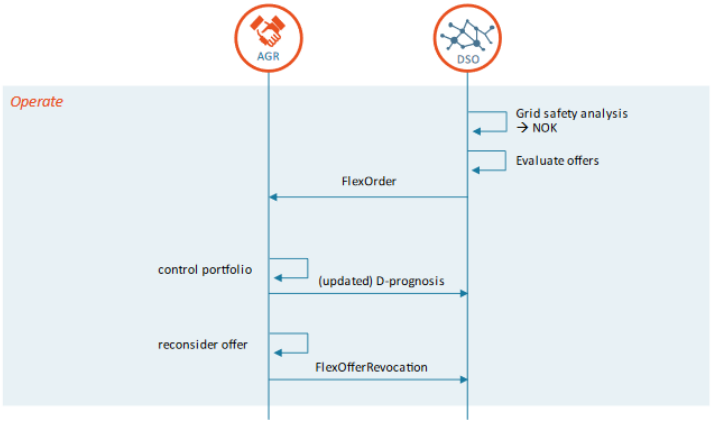


Figure 4.5: UFTP information flow in Operate phase [4]

Finally, the fifth step is the Settlement Phase. At the end of each month, the AGR and DSO calculate the sold and procured flexibility as well as the possible exchanges that were not validated by each other. If a difference arises, then the two parties can dispute. Figure 4.6 shows the UFTP information exchange in the Settle phase.

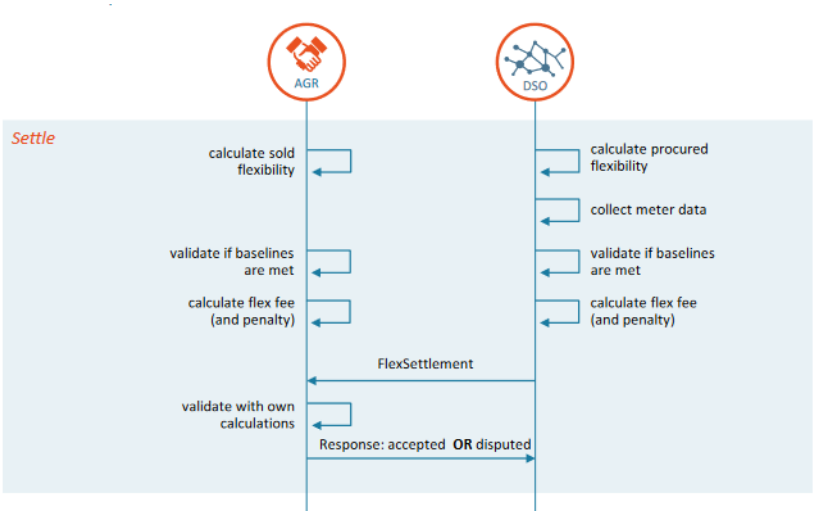


Figure 4.6: UFTP information flow in Settle phase [4]

These are the different steps that the USEF framework utilizes to enable the coordination between DSO and actors that can provide flexibility like an AGR. Additional messages may be sent when bilateral contracts are set up beforehand, but this report will not directly address that. The USEF Foundation has published many different documents to explain the framework. At the time of writing this paper, a clear statement about the granularity of how much power can be exchanged was found in the literature. Two instances were found where the granularity was somewhat mentioned. In the document *USEF: The framework explained 2021*, it is commented that ”Activation is typically expressed in energy (kWh)” [4]. This helps to understand that the magnitude of the exchanges is of kWh but it is unknown whether there is a minimum amount (i.e. steps of 5 kWh). Additionally, in the appendix when defining the flexibility service quantification where it is mentioned: ”Determination of the amount of load/generation reduction/increase in terms of instantaneous power [W] or energy during a certain time interval [Wh].” which seems that flexibility power can be as small as 1 Watt. The other instance when power granularity is mentioned is in an older version from 2015 of the same document *USEF: The framework explained 2015*. The effect of reducing load consumption using a heat pump is explained and the text states ”Reduce peak load in morning hours (5-9) by 1 kW” [24]. With this statement, it would seem that the framework is able to work with power ratings that are as small as 1 kW but because no statement could be found, it will be assumed that the granularity can go down to Watts (i.e. a flexibility request of 3 W).

Despite the fact that GOPACS and USEF seem to do the same task, there are some important distinctions that need to be pointed out.
The first one is the nature of the two. GOPACS is a market mechanism that connects different market platforms together to allow DSO and TSO to coordinate. The USEF framework on the other hand is a communication protocol between the DSO and a party that can offer flexibility. To some extent, it is also a market as the DSO sends requests (demand) and the concerned parties send offers (supply) but it does not use a commercial platform like ETPA like GOPACS does.
The second one is the scale where the two operate. GOPACS allows grid operators to connect and coordinate between themselves to solve congestions while the USEF framework involves actors like aggregators and grid operators. GOPACS therefore works more on a national grid level both for distribution and transmission while USEF is more local, within the distribution level.

4.0.2. PNNL's Transactive energy framework

The Pacific Northwest national laboratory has developed a Transactive energy framework that serves a similar purpose to the USEF Framework. The flexible assets communicate their price-quantity bid curves forecast for the next 48 hours to the DSO. The forecast is based on preferences set by the user who can choose for each individual appliance their sensibility to electricity price. The more sensitive the appliance is, the more it will be willing to change its consumption to moments of low electricity prices. The DSO is also in charge of forecasting the load curve of the non-flexible devices and clearing the market price for a 48h time window [25]. This market-clearing represents the retail price for which the consumers within the DSO network are willing to pay for the electricity. The retail market clearing is then bid in the wholesale market as the energy demand. Thanks to the large time window of 48h, and to the information that flows in, the flexible devices can communicate to the DSO their change in operation when the price is greater than their preference [25]. In the transactive energy framework, the flexible assets are coordinating 24/7 with each other to best fit the demand curve set by the DSO. In the transactive energy framework, all actors in the distribution network are in continuous communication, whether directly or indirectly, to negotiate with each other about energy cost and need [26] [27]. The DSO acts as a bridge between the wholesale and retail markets as shown in figure 4.7. This architecture creates a decentralized and scalable system, where the devices are controlled by the user and not a company, allowing joint market and control to best match demand and supply [27].

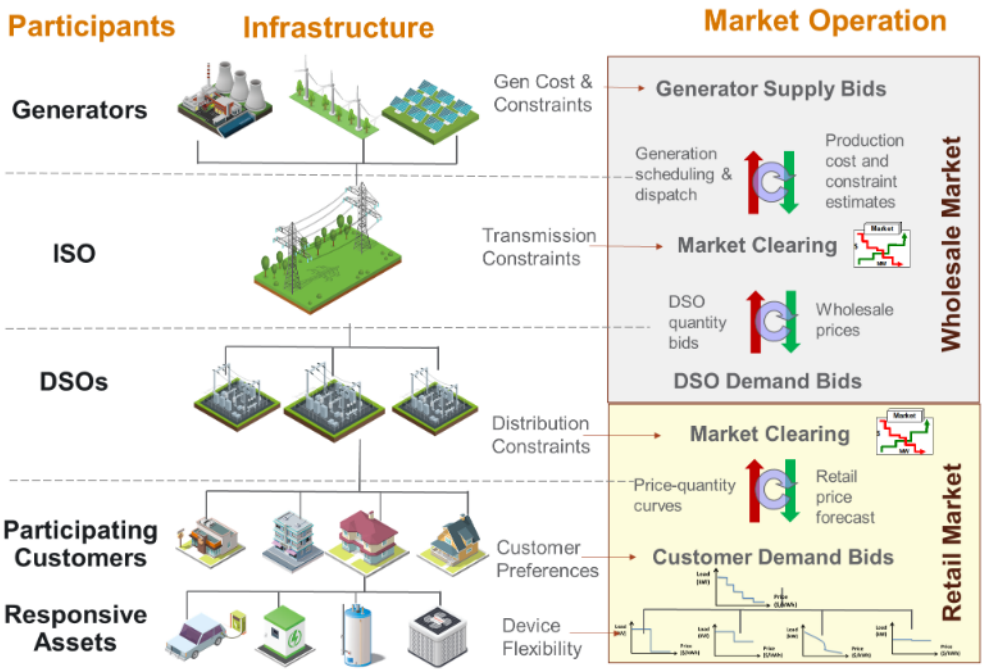


Figure 4.7: Overview of the wholesale and retail market coordination scheme [25].

The study has decided to implement the USEF Framework rather than the PNNL's Transactive Energy Framework. The main reason is because of the greater involvement of the users in the latter framework. As explained, each user needs to set their preferences which will determine the appliance's willingness to participate in the flexibility market. The USEF Framework, on the other hand, does not require that. The AGR can directly talk with the EMS which encompasses all of the user preferences into one single data/message (i.e. the baseline). This characteristic also better suits the EMS being developed for TGV as it was designed to reduce/limit user interaction with the system. Thus, the implementation of the USEF Framework was preferred over the PNNL's solution. Nonetheless, implementing the USEF Framework has some disadvantages. The fact that the baseline for the next 24h is requested can be a problem as depending on the control methodology implemented by the EMS, such a forecast may not

be always available. It is stated in the *USEF Flexibility Trading Protocol Specifications*, that there are alternative baselines methods [23]. The alternative baseline is assumed and agreed upon by both the AGR and DSO. However, as stated by the document, the way that the alternative baseline is computed is out of scope of the USEF Foundation and all documents treat on the default situation described above.

This chapter has introduced and discussed GOPACS and USEF Framework. An in-depth description of the USEF Framework was given with its five steps process. Then, GOPACS and USEF were compared with the main difference being nature, one is a market mechanism while the other is a protocol and the scale at which the two work (national and local respectively). Finally, the PNNL's Transactive energy framework was presented and it was explained why the USEF Framework was preferred over the PNNL's.

5

The AGR architecture and interactions

This section looks more into details about the proposed AGR architecture as well as the interaction that the AGR has with each involved actor (EMS and DSO). First, the AGR program architecture is explained, then the interactions with the DSO and EMS are examined as well as the computation happening within the AGR program such as flexibility bid creating and bid advising. Finally, how the system exchanges information is described by explaining the MQTT messaging protocol and the platform OpenRemote. At the end of this chapter, the reader will have an understanding of the AGR architecture as well as it intrinsic functionalities.

5.1. The AGR program architecture

5.1.1. Achitectural requirements

The AGR program has been designed with some architectural requirements. These have been decided with the intent to make the program as versatile as possible while avoiding any possible error.
The first requirement is EMS interoperability. It is crucial that the AGR program is compatible with as many EMS as possible. By EMS it is meant that the AGR program should be able to interact with different control methodologies (other than MPC, used by the EMS of the TGV). The goal is that the AGR can aggregate with a wide range of EMS which diversifies its portfolio but also increases the capacity of controllable flexibility. To do so, the AGR should take as many responsibilities as possible to unburden the EMS and make it easier to implement the protocol.
The second requirement is redundancy, to avoid communication errors that may lead to unserved flexibility. This is quite crucial for the AGR as an error in planning and serving flexibility means financial penalties. To do so, safety net processes that double-check and confirm decisions are needed.
The third and final requirement is plug and play. As explained previously in chapter 1, the idea of the AGR program is that it can be adapted depending on the AGR operator's wishes. The program architecture must be such that parts can be upgraded or modified to best fit the desired tasks/role. To do so, the program workings must be simple and straightforward to ease the modifications. Additionally, each interaction in the protocol should be clear to users so that when applying modifications or upgrades to the code the protocol message is respected favoring smooth assimilation of changes.

5.1.2. AGR program

The architectural requirements have influenced the way that the AGR program has been designed. To best fulfill conditions, the AGR program is divided into 4 sections each responsible of a different stage of the flexibility bid processing. Figure 5.1 depicts the AGR program architecture division into these four sections.

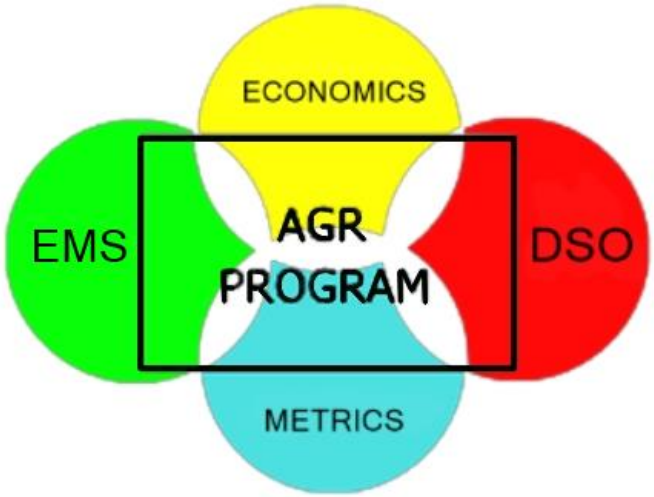


Figure 5.1: AGR program architecture

The four sections' role are:

- The DSO section, responsible for the communication between the AGR and the grid operator. Discussed in more detail in section 5.2.1.
- The Metrics section, responsible for computing the available flexibility and creating flexibility bids. Discussed in more detail in section 5.2.2.
- The Economics section, responsible for setting a price to the bids, bookkeeping the economic transaction, and acting as an advisor in the bidding process. Discussed in more detail in section 5.2.3.
- The EMS section, responsible for the communication between the EMS and the AGR. Discussed in more detail in section 5.2.4.

While the concept is that each section can be interchangeable or modified, it is important to understand they interact with each other and thus all of them are needed for the correct functioning of the system.

5.2. The AGR interaction

This section goes through the interaction that each of the AGR section has with each other and/or the involved actors. The purposes are two: a clear definition of the message protocol and an explanation of each section's role in the flexibility bidding process. At the end of each AGR section, there is a short summary of the data that is generated and shared by the section.

5.2.1. The DSO section (DSO-AGR interaction)

This subsection focuses on the DSO section and the DSO-AGR interaction shown in figure 5.2.

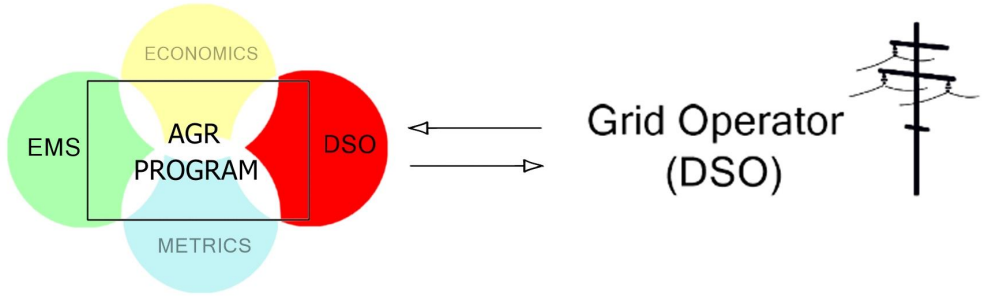


Figure 5.2: The DSO-AGR interaction

The USEF framework has been explained in more detail in chapter 4 and can be summarized into three steps: detection of congestion, request and offering of flexibility, and confirmation of an offer.

Detection of congestion: The purpose of this step is for the DSO to forecast whether the network will be subject to congestion. To do so, it requires every AGR that is connected to a node of the network to send a forecast of the energy program of its grid consumption for the next 24 hours. This forecast is called baseline or D-prognosis in the USEF Framework terminology.
Once the DSO receives all the baseline, it adds them up into one and checks whether the power levels exceed the network limit. This is equivalent to the validate phase in the USEF timeline. Figure 5.3 is a simplified depiction of this process. The DSO can now estimate when the congestion happens as well as the volume of excessive energy congesting the network.

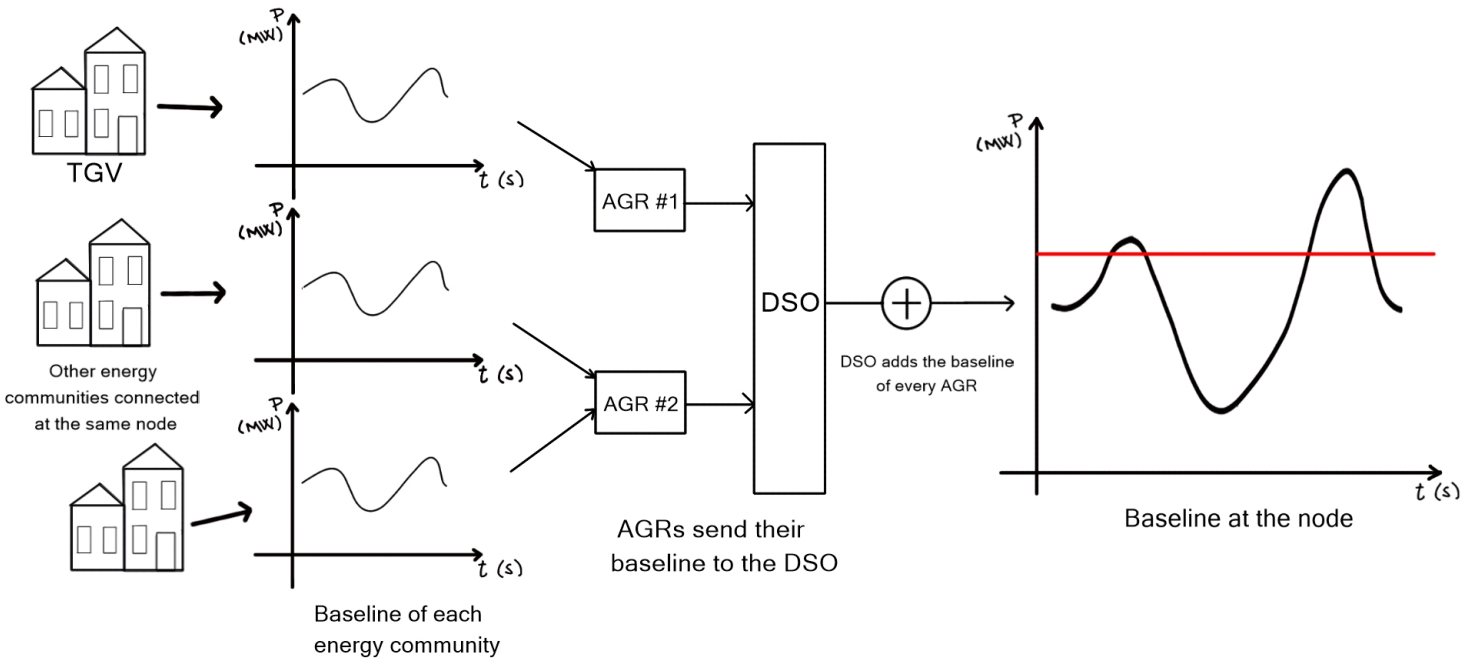


Figure 5.3: Schematic of a congestion detection process

Request and offer of flexibility: Now that the needed amount of flexibility volume at a specific time has been determined, the DSO can send flexibility requests (FlexRequest in USEF). Each request is a volume of power for a specific period. The AGRs at that node can bid on how much of the volume they are willing to match for what price (FlexOffer in USEF). The AGR has two ways of matching the flexibility request of the DSO: either load shedding (or peak shaving) or load shifting.
Load shedding or peak shaving is the reduction of power exchange with the grid. This can be done by either reducing the energy consumption (i.e. an appliance is turned off) or by using local storage. In the case of TGV that would be using a battery and/or fuel cell with hydrogen.
The other option is through load shifting. The consumption is rescheduled during a period when the grid is not under stress. To assist this decision-making, the DSO shares the available energy in the non-congested PTUs. The available energy is the difference between the congestion limit of the network and the combined baseline. If the load shifting option is chosen, then knowing the margin of available energy helps to know which PTUs the load can be shifted without causing another congestion. The build of the AGR program proposed by this paper can only do load shedding and thus load shifting will not be discussed. The reason is that at the time of this research, the EMS was not fully operational and some data was lacking to be able to properly evaluate load shifting possibilities.
Figure 5.4 is the baseline at the node (combined baseline of all loads) with the different possible ways to provide flexibility.

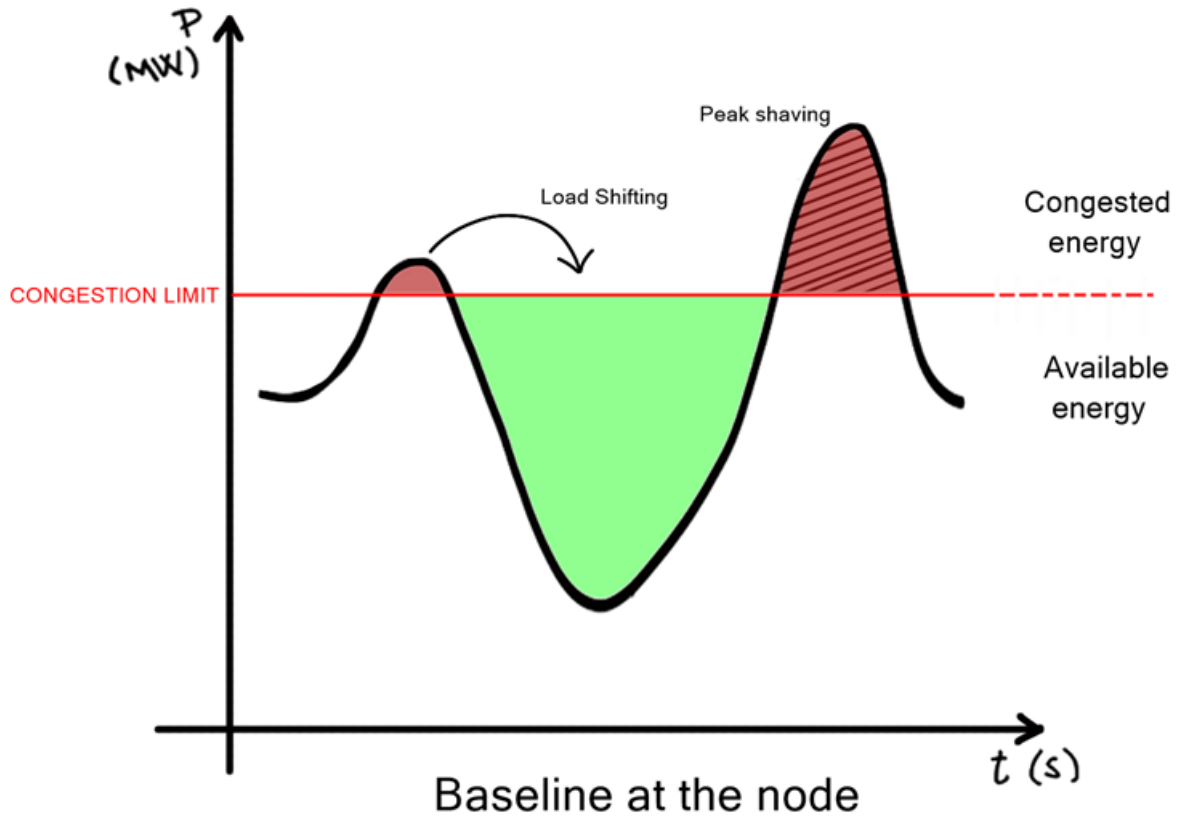


Figure 5.4: Possible way to provide flexibility

Confirming a flexibility offer: Once the DSO has accepted a flexibility offer (FlexOrder in USEF), a confirmation is sent to the AGR who has offered it. Once confirmed, the AGR must send back an energy program which is the baseline modified by the flexibility bids. Once this is done, the DSO expects the actor to respect that energy program otherwise a financial penalty (fine) will be issued. This new energy program is called the DSO Energy program.
Figure 5.5 shows a simplified process of flexibility requested (upper middle figure), offered (lower middle figure) and the confirmation of bids (right-hand figure). In this case, the early morning offer is not accepted while the late evening is.

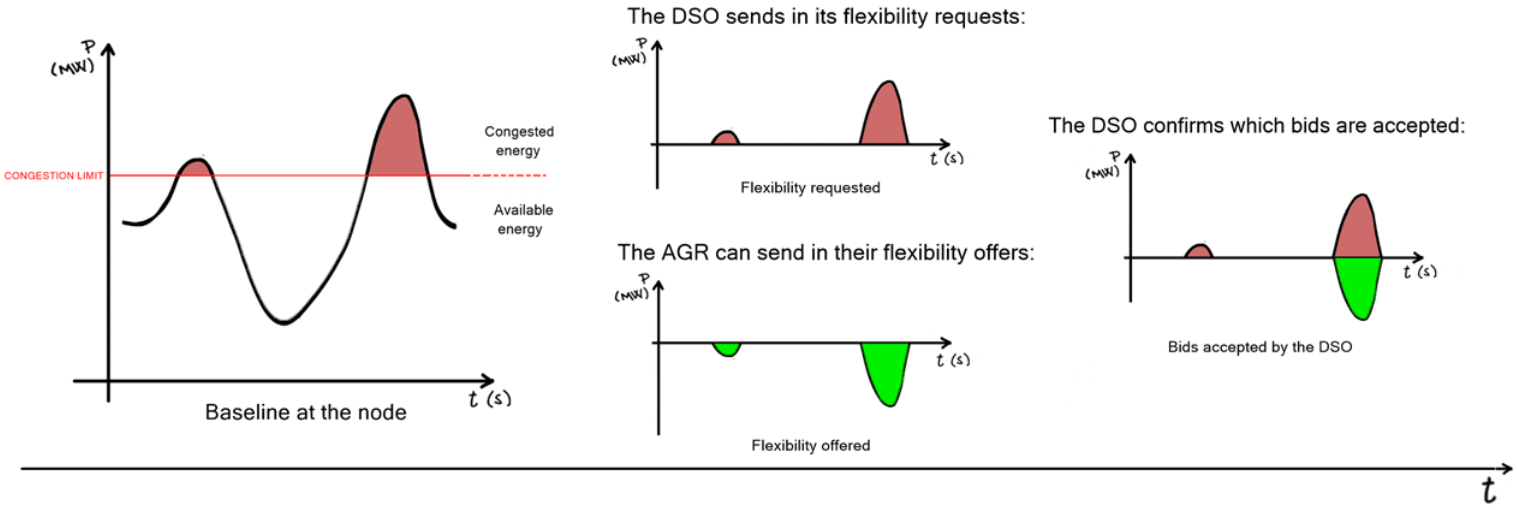


Figure 5.5: Simplified schematic of a flexibility request, offer, and confirmation

From this description of the message protocol between the AGR-DSO, it has been possible to highlight the following message/data:

- A baseline forecast: the energy program for the next 24 hours.
- Flexibility request: the request for flexibility for a certain volume and period.
- Flexibility bids/offers: the offer in volume, price and PTU of flexibility that the AGR is willing to offer.
- Flexibility confirmation: The DSO confirms that it has accepted the bid from the AGR.
- DSO Energy program: the modified baseline after the flexibility changes are taken into consideration. This is the energy program that the DSO expects the AGR to respect with the risk of otherwise being fined.

Figure 5.6 depicts the schematic of the AGR-DSO interaction. It has been described how this interaction enables the forecasting of congestion, the requesting and offering of flexibility, and the confirmation of which bids have been accepted. The next subsection looks into how the AGR program determines the amount of flexibility that the system can provide.

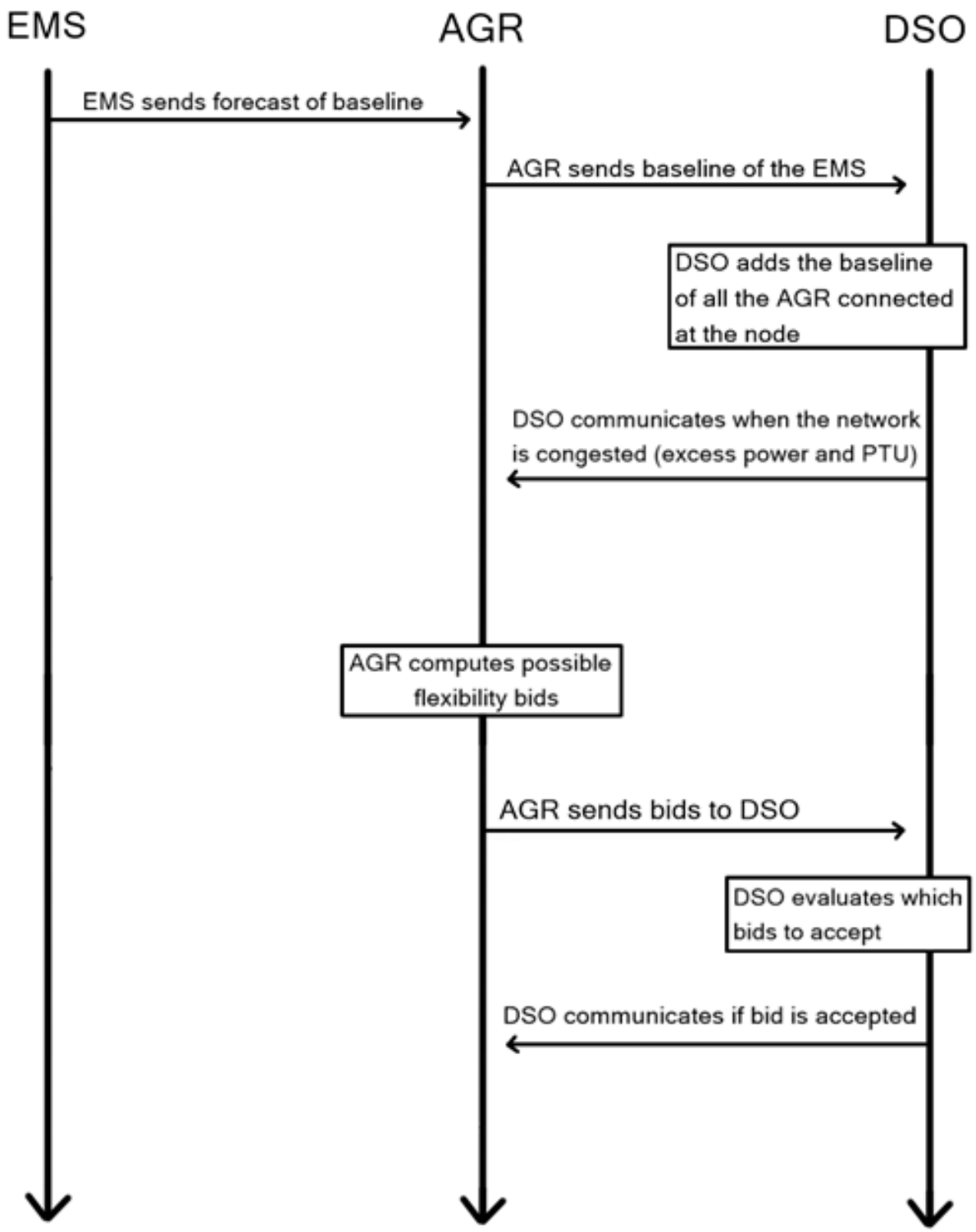


Figure 5.6: Schematic of the AGR-DSO interaction

5.2.2. The Metrics section

This subsection treats on the metrics section shown in figure 5.7 and how available flexibility is determined.

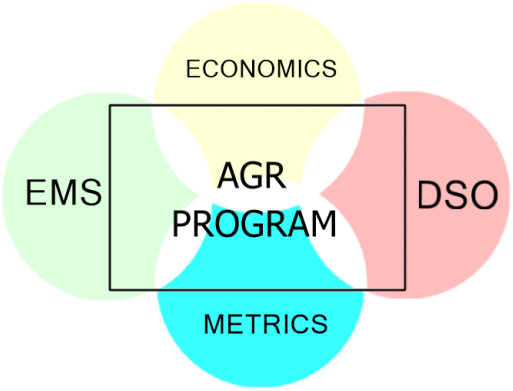


Figure 5.7: AGR program with the Metrics section highlighted

This section of the AGR is reponsible in evaluating the available flexibility and creating flexibility bids or changement in the baseline and planned operation. As explained in subsection 5.2.1, the AGR build proposed by this paper can only do load shedding using the storage asset as power source. Nonetheless, with the correct data, the code can be extended to that functionality too.
The flexibility is provided through the storage asset by discharging it to power the TGV's load and reduce the power exchange with the grid. Such a process is shown in figure 5.8, which depicts the schematic of the simplified determination of available flexibility. The figure is divided into 2 halves, namely step #1 and step #2. Step #1 is the data that the EMS sends to the AGR. This data is comprised of the baseline for the next 24h (shown on the upper quadrant of step #1) and the Battery State of Charge (SoC) for the next 24h (shown on the lower quadrant of step #1). Knowing the SoC for the next 24h, the AGR can estimate how much energy the storage asset will have throughout the day and thus how much it can deliver at a specific moment of the day. Step #2 shows how when the peaks of the baseline are reduced, the SoC level decreases as well.

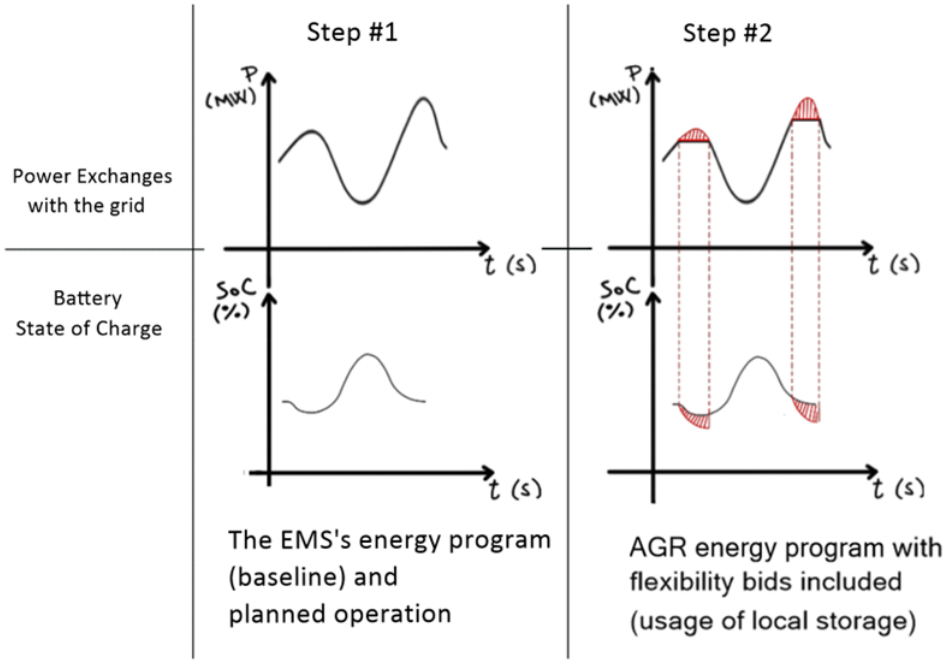


Figure 5.8: Schematic of the simplified process to determine available flexibility

Figure 5.8 shows only the SoC as needed data but in reality, more is being used to determine the available flexibility. Additionally to the SoC, the rated power for the next 24h is shared which allows the AGR to understand how the EMS plans to use the storage asset. For example, if the battery asset has a maximum rated power of 5 kW and the EMS plans to use 3 kW at a moment of needed flexibility. Then the AGR knows that it has only 2 kW available from the battery that it can use to bid flexibility. The SoC and power ratings for the next 24h are part of the set of information named Planned Operation.
The asset specifications should also be known to the AGR so that it can evaluate the available possibilities. For example, the AGR needs to know the capacity of the batteries as the SoC alone is not relevant information. This build of the AGR program is quite static, and the assets do not change. However, the purpose of this program is that it can be dynamic and adaptable. This is useful for an EMS in charge of electric vehicles that are not always plugged in and available for trading. For this reason, a dictionary containing all the various information should be given to the AGR. This dictionary should be continuously sent to the AGR as the assets' availability may differ (i.e. EV plugging in and out). Figure 5.9 shows what the structure of said dictionary would look like. Please note that the parameters are not the real ones of the assets, the cases were filled up just to give an example.

Asset	Outputs E (I/O)	Storage (I/O)	Capacity (kWh)	Rated Power (kW)	Range of Rated Power (%)	Ramp up to rated power (MW/h)	Available (I/O)
Battery	1	1	50	5	0 -100	15	1
Fuel cell	1	1	100	10	0-100	15	1
Heat Pump	0	0	-	5	0-100	5	0
PV System	1	0	-	25	0 – 100	-	0

Figure 5.9: Example of a dictionary structure

The explanation of each column is:

- Outputs E: Allows to know whether the asset generates energy or not. This is helpful to understand whether the congestion will be resolved by using local generation or if it will be by reducing consumption.
- Storage: Allows to know whether the assets can intake and store energy.
- Capacity: Capacity of the asset.
- Rated Power: Rated power of the asset.
- Range of Rated Power: How much the rated power can change. This is helpful to know whether the asset has the functionality of modulating its output. An example of utility is that it can be used to slowly ramp up and match the flexibility demand curve.
- Ramp up to rated power: Gives information on how fast the ramping is up of the devices is. Helps to understand how to compute the cumulative flexibility curve (IPP summation) [12].
- Available: Whether the asset can be used for flexibility or not. This is useful for a system that has assets under maintenance and that are no longer active or for assets that may be active only for a certain moment of the day (EV gets plugged in) or for the AGR to have an overview of what the system looks like.

Now that the data available to the AGR has been explained and discussed, it is time to look into the way that it determines the available flexibility.
As the possible feasible flexibility varies based on many different factors an optimization problem has been created to compute how much the AGR program can offer to the DSO. Equation 5.1 shows the objective function of the optimization.

$$objective\ function = min(FR_i - \sum_j (P_{sh_{ij}})) \tag{5.1}$$

Constrains:

$$P_{rated_{min_j}} \leq P_{sh_{ij}} \leq P_{rated_{max_j}} \tag{5.2}$$

$$P_{sh_{ij}} \leq \frac{Cap_j \cdot SoC_{ij}}{4} \tag{5.3}$$

$$P_{sh_{ij}} \leq P_{rated_{max_j}} - P_{rated_{ij}} \tag{5.4}$$

$$0 \leq P_{rated_{i,batt}} \tag{5.5}$$

with:

- FR the flexibility request (kW)
- $P_{rated_{max}}$ the maximum rated power of the asset (kW)
- $P_{rated_{min}}$ the minimum rated power of the asset (kW)
- P_{sh} the flexibility power the AGR provides (kW)
- Cap the capacity of the asset (kWh)
- SoC the State of Charge of the asset (%)
- i index of the PTU
- j index of the asset

The goal of the AGR is to match the flexibility power requests (FR) at every PTU by using the power of local storage of its different assets (Psh).
Thus, the objective function of this optimization problem is a minimization of the FR and Psh with the indexes i and j, the PTU and assets index respectively. At every PTU the AGR program will try to match as much as it can the flexibility request without considering users' preferences. The outcome of the optimization is a new energy program called AGR energy program (modified baseline).

As explained, the system is physically constrained by the asset's limitation as well as the planned operation. Let us break down each of the constraints shown in equation 5.2 to 5.5.
The first constraint, shown in equation 5.2, makes sure that the power remains within the maximum and minimum limit of the assets. With this constraint, the AGR is sure to use the asset within its power limits and not bid more power than it can physically deliver.
The second constraint, shown in equation 5.3, relates to the available power. By using the capacity (Cap) of the asset and the state of charge (SoC) it is possible to compute the amount of energy that the storage has and thus the amount of power that it could possibly deliver. The reason why the capacity and SoC are divided by 4 is that each data point represents 15 minutes of operation.
The third constraint, shown in equation 5.3, relates to the amount of rated power (P_{rated}) being used. If the asset is already being used to power another load, the AGR must know how much power is "left" to use before the asset reaches its maximum rated power. This prevents the AGR to make a flexible bid without considering the consumption already happening in the background.
The fourth constraint, shown in equation 5.5, checks that the rated power of the battery is not negative. If that was the case then the battery could not be discharged. This is not a problem for the hydrogen storage as the charging and discharging are decoupled.

There is a limitation with the versatility of the optimization that needs to be addressed. As explained the point of the AGR is to be to adapt to different assets. The concept of the dictionary was introduced. While receiving information is not a problem for the program, it must be linked to an asset that was

already defined in the optimization program. To give a more concrete example, if the optimization was built to work with one battery and one hydrogen storage, through the dictionary, the capacity, max rated power or availability could be changed in case the assets would be modified (i.e. more hydrogen tanks are installed and the capacity increases). So in that sense, the dictionary is doing its job correctly as the system specification are not static. However, if an additional flexible asset would be installed, for example an EV, then the dictionary alone would not be able to automatically add an decision variable in the optimization algorithm. Thus the AGR program can be versatile only if the installed assets are known beforehand and the optimization function has been modified to include them.
It has also been explained before that the AGR program is unable to compute load-shifting flexibility bids. The reason is the lack of information available from the state of the EMS of TGV at the moment. Nonetheless, it is quite an important aspect as most EMS that the AGR interacts with are unlikely to be equipped with a large capacity storage asset like TGV thus not being able to provide flexibility like these simulation cases. Load shifting is then a resourceful action that should not be ignored in future iterations of the AGR program.

This optimization problem has been modeled using the Pyomo library. Please refer to the appendix B for more detail about the code.

From this description of the metrics section, it has been possible to highlight the following message/-data:

- The Planned Operation: how the asset operation has been planned for the next 24h. This includes the variation in SoC and rated power.
- Dictionary: all relevant data of the assets. Ranging from physical specifications to the availability of the asset.
- AGR energy program: the new grid exchange computed by the AGR program after the optimization algorithm.

5.2.3. Economics Section

This subsection discusses the Economics section highlighted in figure 5.10 and its role in flexibility bidding.

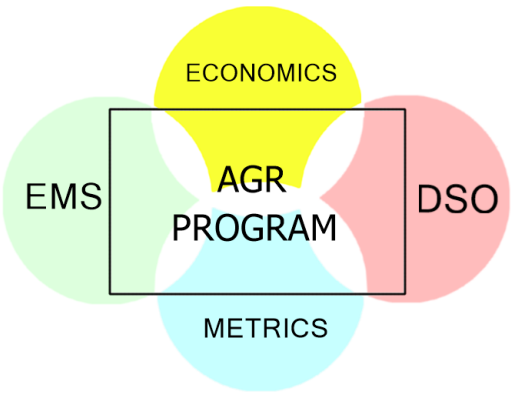


Figure 5.10: AGR program with the Economics section highlighted

The economic section has a variety of roles, each with a different level of importance. The idea behind this section is that it acts as a price setter, bookkeeper as well as advisor to the AGR operator.

Its responsibilities range from common financial tasks such as determining the price of each bid to calculating the revenue and losses based on the served and unserved (offered and accepted bids that have not been served) flexibility. Other tasks are, ranking bids or creating ratios to more easily extrapolate and understand information about the system's progress.
While determining the price is an essential task as the AGR needs it when communicating its bids to the DSO, the others are less fundamental but can still have a crucial role. This section explains each of them and discusses their function.

The first task is price setting. The Economics section makes sure to determine an appropriate price for each flexibility bid depending on which market time horizon the AGR is bidding on (day ahead or intraday). The AGR can participate in different markets depending on what kind of service it wants to provide. In this report, it was decided that the AGR would only operate in the balancing market and thus the price should be determined with respect to that market [4].
This report didn't dive into the different methods of forecasting but having a reliable prediction of the market price can be of great advantage for the AGR for two reasons:

- The AGR can correctly price the flexibility it sells to win the flexibility bidding process and maximize revenue.
- The AGR can plan its operation in the coming hours/days. Deciding to hold its bids to the next day because it forecasted greater prices.

Overall, the price setting is essential not only because it is required by the AGR to assign one for each bid to participate in the USEF Framework but because it can increase the efficiency of the bidding process and help plan the operation.

The second task is revenue and losses. These are quite straightforward, it is the multiplication of the volume of power times the price (or penalty) at which it has been sold. There can be two types of revenue and loss, namely the bided revenue and loss and the served revenue and unserved loss. There is a clear difference between these pairs that needs to be understood.
The first one, bid revenue/loss, refers to the future monetary outcome of selling or not bided flexibility. This means that a volume of power at a certain price is bid to the DSO and still awaiting a response on whether it has been accepted or not. The bid loss is a bid that was initially created by the AGR (AGR Energy program) but then rejected which resulted in a loss in possible revenue.
The second pair, the served revenue and unserved loss refers to the bids that have been accepted by the DSO and that the AGR has served or not. Serving the planned flexibility results in financial compensation as the AGR has performed the action that was agreed upon (i.e. reducing its own energy consumption). However, as explained in the first section, there can be instances when the AGR does not provide flexibility due to complications. If that is the case, then a penalty needs to be paid hence the name unserved loss.
It is important to state that these two pairs can apply to both day-ahead and intraday market bids.
The relevance of having such metrics is two. The more obvious one is as a bookkeeping function. Using the served revenue and unserved losses the AGR can keep track of whether its is making profit or not. The other pair, the bid revenue and loss, has more of an advisory purpose. The more basic intent would be to use the bid revenue to have an idea of how much the set of bids will yield if accepted. Another function would be to compare two sets of bids. For example, if the AGR's proposed bids are partially refused then the bid revenue and loss can help examine how much of the AGR's financial plans have been altered. Overall, they help to monitor how the AGR financial returns.

This leads to the third task, creating a rank. The concept is that it grades each bid based on the revenue it brings in compared to the maximum revenue possible out of the set of bids. The equation 5.6 shows how it is calculated with $Revenue$, the revenue that a bid generates at index i (PTU) and $Revenue_{max}$ the maximum revenue that is achievable out of the set of bids analyzed.

$$Rank_i = \frac{Revenue_i}{Revenue_{max}} \tag{5.6}$$

The rank permits to organize hierarchically each bid based on its importance. This can be of great use in a situation when a decision must be made for which bid needs to be kept and which needs to be discarded. It will be demonstrated in section 6.2.4 on how the ranking of bids can have a significant impact.
The rank has the purpose of both advising and monitoring. Advising because it helps the AGR filter the bids based on their importance as explained before. Monitoring because the AGR can examine its performance by keeping track of the ranks of the bids that were either not secured (bid was not accepted by DSO) or not provided (the flexibility was not served resulting in a penalty). This can help the AGR operator to investigate and adjust its bidding strategies.

The fourth task is creating ratios. Similarly to the ranks, the ratios have the intent of monitoring the AGR performance by using a variety of different measurements. For example, a ratio can be done between the unserved power and the served power as shown in equation 5.7

$$Ratio_{served} = \frac{\sum Unserved\,Power}{\sum Served\,Power} \tag{5.7}$$

With the ratio, the AGR operator can monitor how its system is performing. If the ratio becomes greater than one, then it means that more power has been unserved, thus the system is underperforming. The choice of the timespan that the ratio should cover is up to the operator's choice. For example, the operator can choose the timespan of one calendar day. Then at the end of the day, the operator will have an overview of how the system has performed. It is up to the operator to decide which combination to use, depending on which one it finds more suitable.
Here are two other combinations proposed:

- Unserved intraday vs Unserved day ahead: ratio between the unserved power offered in the intraday market and the unserved power offered in the day ahead. This helps the AGR operator to monitor how the bidding system performs for the two markets. If the ratio becomes large (i.e. greater than 1.5 or smaller than 0.5), then it signals the AGR operator that something along the bidding and serving of flexibility is going incorrectly.
- Losses vs Revenue: ratio between the losses and revenue. Depending on the timespan, it helps the AGR operator to determine whether the profit margin is positive or negative. For example, if the ratio is greater than one (the financial losses are greater than the revenue) then the AGR can decide to bid in a more conservative way to avoid further possible losses, or alternatively bid in a more aggressive way to recover the costs.

The Ratio was not envisioned as a metric to activate an automated response from the AGR program but rather to signal the AGR operator. Once the operator has received a signal, it can act according to what are his plan. However, if the AGR operator clearly defines the course actions that it wants the program to follow when receiving a certain ratio's signal then the program can be automated.
It is also important to note that some of the ratio combinations can lead to similar metrics introduced in section 5.2.2. In fact, both the ratio and rank were created using the same goal set in the paper *Quantification of operational flexibility from a portfolio of flexible energy resources* [12] and that is to make metrics that are intuitive. Through ratios, it is easier to understand the weight of a decision rather than using an absolute number (i.e. all ranks below 50 €are rejected vs all bids with rank below 0.5 are rejected).
The choice of the rank and ratio is quite important as chosing the correct one is essential. However, before explaining how the correct rank is determined and how the ratio affects this outcome, more aspects of the protocol need to be introduced. The determination of an optimal rank is further explained in section 5.2.6, once the whole protocol has been explained.

From this description of the economic section, it has been possible to highlight the following message/data:

- Bid pricing: the forecasted price that should be set for each bid.
- Revenue and Loss: the income and losses due to served and unserved flexibility.
- Ranking and Ratio: metrics used for monitoring and advising the bidding process.

Please note that only the first bullet point "bid pricing" is necessary for the correct functioning of the protocol. The others are just additional data that can serve as aid.

5.2.4. The EMS section (EMS-AGR interaction)

This subsection discusses the EMS section and the EMS-AGR interaction highlighted in figure 5.11.

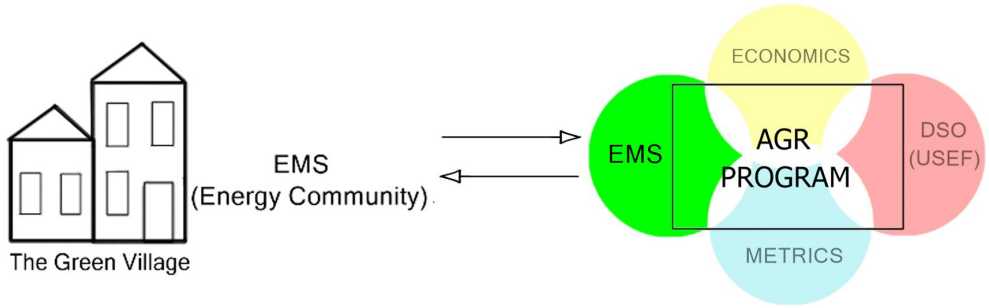


Figure 5.11: The EMS-AGR interaction

This interaction has been chosen to be discussed for last because many of the involved building blocks have been defined in the previous sections. Thus, it has been considered easier for the reader to understand it if all previous steps were explained. The previous subsections have been referenced throughout the text so that the reader can easily go back and clear any confusion.

The role of the EMS-AGR interaction are two. The first one is to share data with the AGR such that it can correctly evaluate the available flexibility and the second is to make sure that the flexibility bids created by the AGR are feasible.

As explained in subsection 5.2.2, the AGR will take care of evaluating the flexibility bids based on the information given by the EMS. Through an optimization algorithm, the AGR program creates flexibility bids that change the inital baseline forecasted by the EMS, called AGR Energy program. This energy program tries to fully match the flexibility request within the system physical limitation. This means that the users' preferences are not taken in consideration and economic yield is maximized. This leads to the need of some sort of check that makes sure that the AGR Energy program somehow does not undermine the user. This is where the EMS comes into play.
The EMS, responsible for how to implement the flexibility, will analyze the AGR Energy program and decide whether it is feasible to implement the bids. If accepted, the EMS will send the same energy program back to the AGR. Elsewise, a redacted version of the AGR Energy program is sent back. The AGR will then know which of the initially computed bids are valid and which ones were rejected. The program that is sent back by the EMS (either redacted or the same) is called Final Energy program.
tthe AGR will send the Final Energy program to the DSO in the form of an offer and the DSO confirms which bids it accepts by sending an energy program back. This energy program sent by the DSO is called DSO Energy program. The AGR will send it to the EMS to confirm which bids have been accepted and how the baseline needs to be changed. The DSO Energy program can be the same as the Final Energy program or parts of it as the DSO is not obliged to accept all flexibility bids offered by the AGR. In case none of the bids are not accepted, then the AGR can compute new bids which will have to go through the same process again. Figure 5.12 depicts the decision-making process to check the feasibility of the bids.

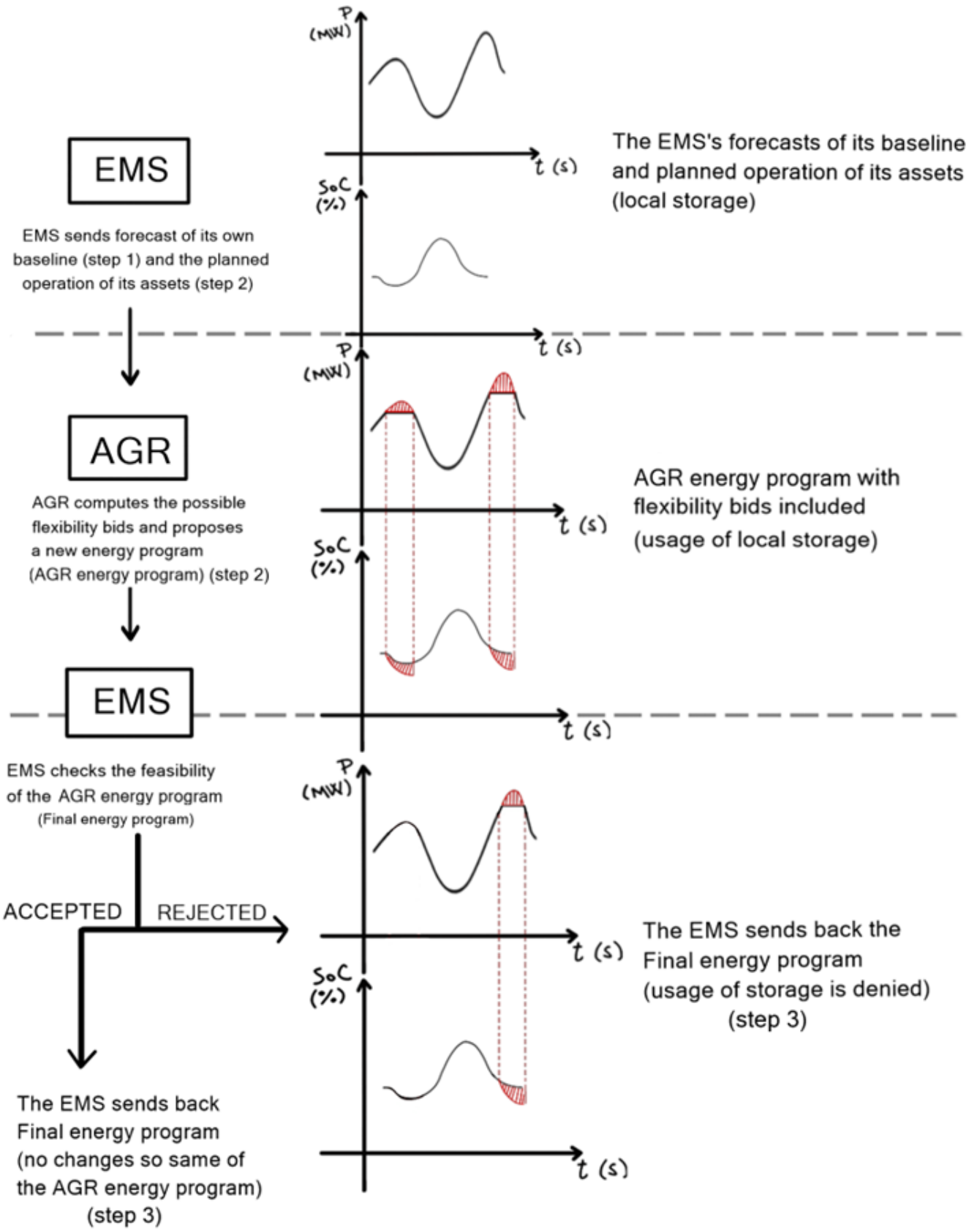


Figure 5.12: Decision-making process for checking flexibility bids feasibility

Now that a general overview of the timeline and how the interaction takes place, it is time to look more closely at each steps that are denoted in figure 5.12.

The first step is receiving a baseline. The EMS must be able to give a prediction on how its grid consumption will be like. As explained in 5.2.1, it covers a full calendar day (24h) as the AGR needs to send it to the DSO so that it can be determined whether there will be any congestion at the node. It is quite important that the grid load forecast is correct as a deviation from the planned program can result in a fine [4].

The second step is determining the available flexibility. As explained in 5.2.2, the EMS provides data that will allow the AGR to determine how much flexibility is available. For example if flexibility is provided by load shedding using storage, the state of charge and power rating of the storage assets at every PTU is needed, as the AGR will know the possible changes in planned operation that it can do. It is in this step that the AGR Energy program is created.

The third step is checking whether the AGR Energy program is feasible or not. It is up to the EMS to decide what can and cannot be achievable. Once done the EMS sends what is called the Final Energy program. The EMS can either resend the same energy program that was proposed by the AGR (if all flex bids have been accepted) or a modified energy program (if the EMS redeems certain bids unachievable). The EMS can modify the bids by reducing the offered volume either partially or fully.
For example, if the AGR wants to reduce an exchange with the grid by using local storage or load shedding, then the EMS can only reduce the volume of flexibility offered. This will result in either less volume of local storage used, less volume load shed or less volume rescheduled. This can be visualized in figure 5.12 at the bottom right part when the EMS has rejected the AGR Energy program. As it can be seen the usage of storage in the morning peak has been denied.

It is important to also note that when the AGR Energy program is accepted by the EMS and all the bids have been accepted by the DSO, the EMS can choose how to deploy/implement the program however it wants, until the energy program is respected. It does not matter if the flexibility was originally planned to be provided through the battery, but the EMS uses the fuel cell or does load shedding. The AGR should have no final saying on this matter. All it matters is that the energy program agreed by the DSO is respected.

Figure 5.13 lists all the messages that are outputted by the EMS (and sent to the AGR) and inputted to the EMS (sent by the AGR). Please not that the DSO Energy program has been included although not shown in figure 5.12.

From this description of the EMS-AGR section, it has been possible to highlight the following message/-data:

- Final Energy program: the redacted AGR Energy program after the feasibility check done by the EMS.

The reason is only one message is because all the other messages were already defined in the previous interactions.

Message	Information	Goal	Timeline	Step
Output	**(EMS to AGR)**			
Baseline	Full day ahead forecast of the grid load consumption.	The grid consumption forecast needs to be sent to every day so that the DSO can forecast possible congestions.	The latest updated version of the baseline until the AGR sends it to the DSO. After this, the energy program must be respected as close as possible.	Step 1
Data	Planned operation of the assets (i.e. SoC & power ratings over the 24h).	Allows the AGR to compute how much flexibility is available to bid to the DSO.	Every 15 minute.	Step 2
Dictionary	Lists of all the assets present in the system as well as their specifications (i.e. maximum power rating, capacity).	Allows the AGR to understand what kind of assets is available	Every 15 minute.	Step 2
Final Energy Program	Either the same energy program that was initially proposed by the AGR or a modified version.	Tells the AGR what the EMS can do. If the energy program is the same as the one sent by the AGR then it has been fully accepted. Otherwise, the AGR needs to correct its bids.	Every time the AGR sends an energy program that is different than the baseline.	Step 3
Input	**(AGR to EMS)**			
AGR Energy Program	The new energy program. It is the modification of the baseline and the changes in the assets planned operation.	This is sent by the AGR, and it is supposed to represent the changes in the baseline and asset operation if the flexibility bids were to be accepted by the DSO.	Every time the AGR wants to make a bid to the DSO's request.	Step 2
DSO Energy program	The energy program that has been accepted by the DSO.	The confirming to the EMS which of the offered bids have been accepted. This way the EMS knows what energy program it must follow.	Every time the AGR makes a bid, it will receive an answer from the DSO.	-

Figure 5.13: Message list for the EMS-AGR interaction

5.2.5. The complete system interaction

Each interactions have been defined and explained, this next subsection ties them all together to have the overlook of all interactions and their timeline.

The system interaction can be subdivided into 3 sections shown in figure 5.15

- Fexibility request: where the amount of flexibility needed is determined (arrow 1 to 3).
- Flexibility bids creation: where the available flexibility is determined and the new energy program is created (arrow 4 to 6).
- Flexibility bids approval: where the flexibility bids are sent to the market and approved by the DSO (arrow 7 to 9).

Figure 5.14 shows the message lists of the whole system interaction. Each message is color graded depending on which AGR section is involved. Green for the EMS section (message from EMS to AGR or vice versa), red for the DSO section (message from DSO to AGR or vice versa). The numbering of each message corresponds to the numbering of each arrow shown in figure 5.15.

#	Name	Time	Type	Content	Description
1	Baseline	Every 15min.	Dataframe	Power exchanged with the grid and local consumption for every 15min of the next 24h.	The energy program forecasted by the EMS for the next 24h. This is all grid exchange between the DSO and TGV.
2	USEF Energy program	Any time before market gate closure.	XML	Baseline submitted by the AGR to the DSO.	The AGR submits the energy program forecasted by the EMS (baseline) to the DSO. Every AGR connected at a node and willing to participate in the local flex market needs to do this.
3	USEF network congestion	At the market gate closure.	XML	Excess power at every PTU. Available power at every PTU.	The DSO has forecasted future congestions and requests an amount of power at each PTU. The AGR can now bid for how much he is willing to activate his flexibility.
4	EMS planned operation	Every 15min.	Dataframe	SoC (if applicable) and rated powers of the assets, local consumption.	The EMS shares its planned operation for the next 24h. With this information the AGR can run an optimization and modify the baseline by changing the planned operation and local consumption.
5	AGR Energy Program	Any time before gate closure.	Dataframe	Energy program (modified baseline) + changes in planned operation.	The AGR sends a proposed energy program that includes the bids. (modification in baseline, planned operation and local consumption).
6	Final Energy Program	Once after receiving the AGR Energy program.	Dataframe	Corrected energy program (modified baseline) + changes in planned operation.	The EMS checks the feasibility of the AGR Energy program and modifies it if needed. If the EMS sends back the same energy program as the AGR's one, then the EMS has accepted all bids/proposed modifications.
7	AGR bids	Any time before gate closure.	XML	Energy program + bids (power, price for the PTU).	The AGR sends the Final Energy program back to the DSO together with the bids. Each bid contains the amount of power at every PTU that the AGR is willing to deviate from the submitted baseline. The financial compensation for each deviation is also included.
8	DSO Energy program	Once after receiving the AGR bids.	XML	Confirmation message with the energy program that the AGR should follow.	The DSO confirms with the AGR which bids are accepted and which are not.
9	AGR confirmation	Once after receiving the DSO bids response.	Dataframe	Energy program (modified baseline) + changes in planned operation.	The AGR makes modifications to the Final Energy program by including only the bids that have been accepted by the DSO and sends it back to the EMS.

Figure 5.14: Message list for the complete AGR interaction shown in figure 5.15

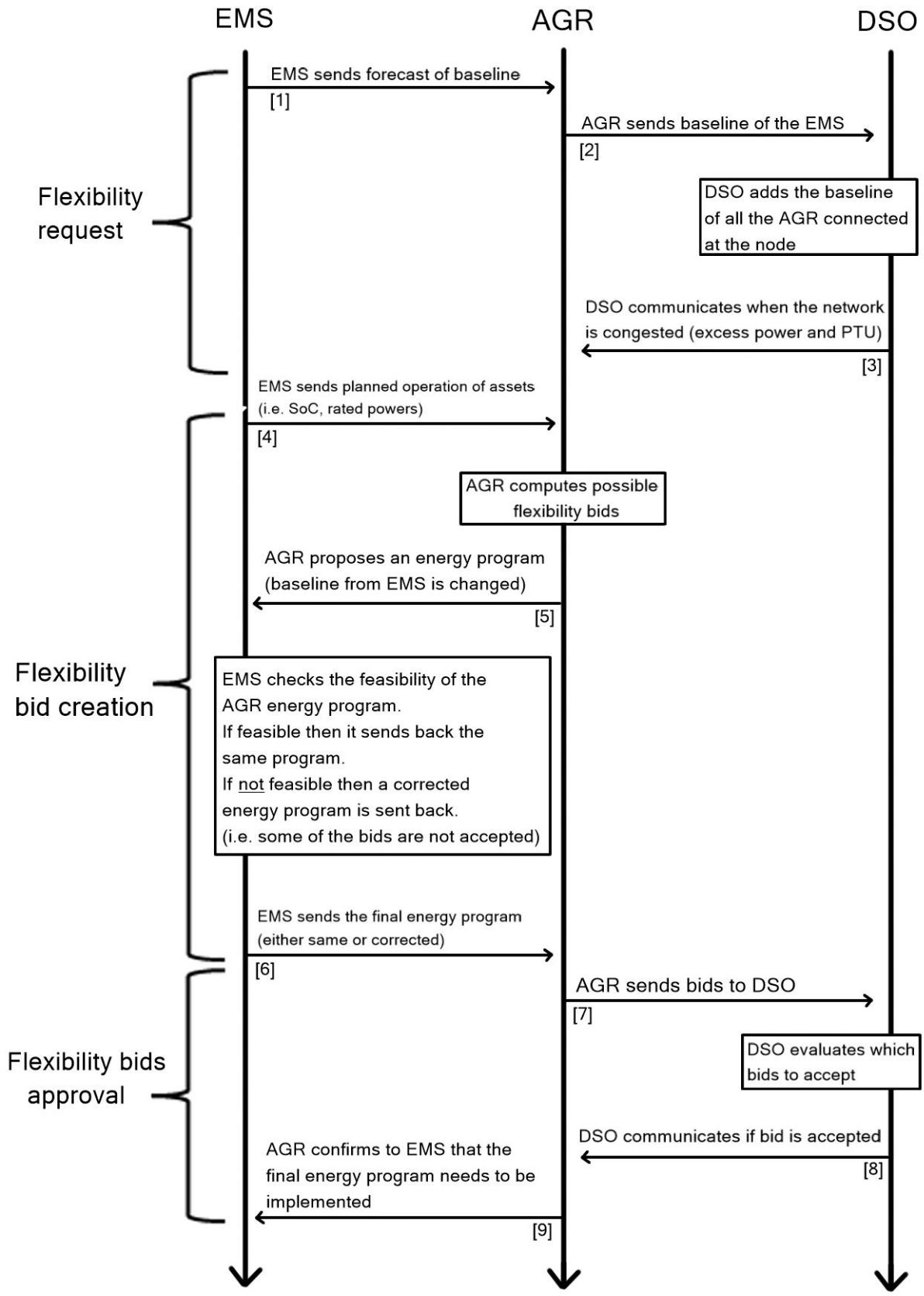


Figure 5.15: Schematic of the complete system interaction.

5.2.6. Rank implementation in protocol

Now that every step of the protocol has been explained, the choice of the rank to use can be explained. The reason why this section is done separately is because of two reasons. The first one is that the usage of rank filtering and ratios is not essential to the operation of the protocol. This means that the protocol can be run without implementing it. The second is because the fundamental structure of the protocol needs to be understood for the rank filtering to make sense.

Rank ratio choice

Filtering through ranking must be done carefully because not all filtering can yield to an advantageous situation. If the bids are filtered using a rank of high value (i.e. 0.9), then most of the bids will be filtered and the AGR will not serve much energy. While the amount of energy served is less (pleasing the EMS preferences), the revenue is most likely smaller than the Final energy program proposed by the EMS. It makes thus no sense for the AGR to choose the energy program filtered by a high rank as it has a smaller revenue compared to the EMS's correction. In that case, the AGR is better off accepting the correction of the EMS.
The same logic can be applied but this time with a rank that is too low (i.e. 0.2). In that case, the revenue will be high (as many of the original bids computed in the metric section are kept) but too much energy is served compared to the energy program after the feasibility check. The EMS is likely to refuse such an energy program as it requires too much energy and exposes users' comfort.

Determining the optimal Rank for filtering and proposing the filtered energy program

As explained above, it is clear that choosing the correct rank for the filtering is essential. Doing it manually is not suitable as it can be lengthy especially if many different sets of bids needs to be checked. It is thus beneficial to have an automated way to compute it.
The procedure is that the AGR would receive the final energy program from the EMS (AGR energy program after feasibility check), then the AGR would need to compute the optimal rank and do the filtering accordingly and then propose this rank-filtered energy program to the EMS again to see if it is feasible. Figure 5.16 shows the system schematics (also previously presented in figure 5.15) with the addition of where the rank filtering process shown in red. A loop is then created in the 'Flexibility bid creation' section of the protocol.
When the term optimal rank is used it is meant as the rank that satisfies both the EMS preferences and the AGR preference. That is, higher revenue and lesser served energy than the final energy program computed by the EMS. To find the optimal rank, an automated script has been created named 'Ranking_Automated.py' which checks whether the rank-filtered energy program has a greater revenue than the final energy program while also serving less energy. This condition can be expressed as ratios using the equation 5.7 introduced in section 5.2.3. The ratio that expresses the requirements can be written as such:

$$Ratio_{power} = \frac{Power\ served\ Unranked}{Power\ served\ Ranked} \leq 1 \tag{5.8}$$

and

$$Ratio_{revenue} = \frac{Revenue\ Unranked}{Revenue\ Ranked} \geq 1 \tag{5.9}$$

With $Power\ served\ Unranked$ and $Power\ served\ Ranked$ the power served by the final (EMS) and rank filtered energy program, respectively. While $Revenue\ Unranked$ and $Revenue\ Ranked$ are the revenue of the final (EMS) and rank-filtered energy program respectively.
The outcome of equation 5.8 needs to be equal or smaller than 1 as the filtered energy program needs to offer either the same amount or less than what the final energy program. On the other hand, the outcome of equation 5.9 needs to be equal or greater than 1 as, if the filtered energy program does not yield a greater revenue, then the AGR should accept the correction done by the EMS during the initial feasibility check.

To automate the process, this analysis is encompassed in a while loop that increments the rank of the filtering by a chosen step until either the conditions are fulfilled or all the ranks have been tested (i.e. the rank is equal to one, there is no more possible iteration).
Three aspects of this method need to be discussed. The first one is how much the ratio is increased every time. If at every iteration the rank is incremented by a large step (i.e. 0.1 or more), then it is more likely that a suboptimal or no optimal rank is found. The reason is that the smaller is step size is, the more likely the rank found will closely meet the conditions 5.8 and 5.9. A setback with a smaller rank increment is that it takes more time for the script to determine it. While this is not a huge delay when communicating with one EMS, the latency increase may be perceived when more EMS are involved and the AGR needs to find an optimal rank for each of them.
The second aspect is how the AGR communicates with the EMS. At the moment the filtering through ranking is presented as one interaction, meaning that the AGR determines the optimal rank, filters the bids with that rank, and proposes the filtered energy program to the EMS. However, the AGR could also just slowly filter the bids, and propose the filtered energy program to the EMS at every iteration until it reaches the optimal rank. The intention behind this method is that the AGR losses as few bids as possible. It is likely that the EMS will refuse those energy programs as they would serve more energy than the EMS would prefer but more testing (with an operational EMS) is needed to rule it out. However, the problem of latency explained previously becomes more severe as more interactions are needed.
Finally, the aspect is that filtering through ranking emphasizes the usage of assets at moments of peak consumption. While the overall amount of served energy flexibility is reduced, the EMS may need that energy the most at those moments. The whole rank analysis is done without considering the time value of energy, quite a vital aspect for an EMS as it does not want to run out of energy when the users are more likely to consume. More testing with an operational EMS is needed to determine the effect of this.

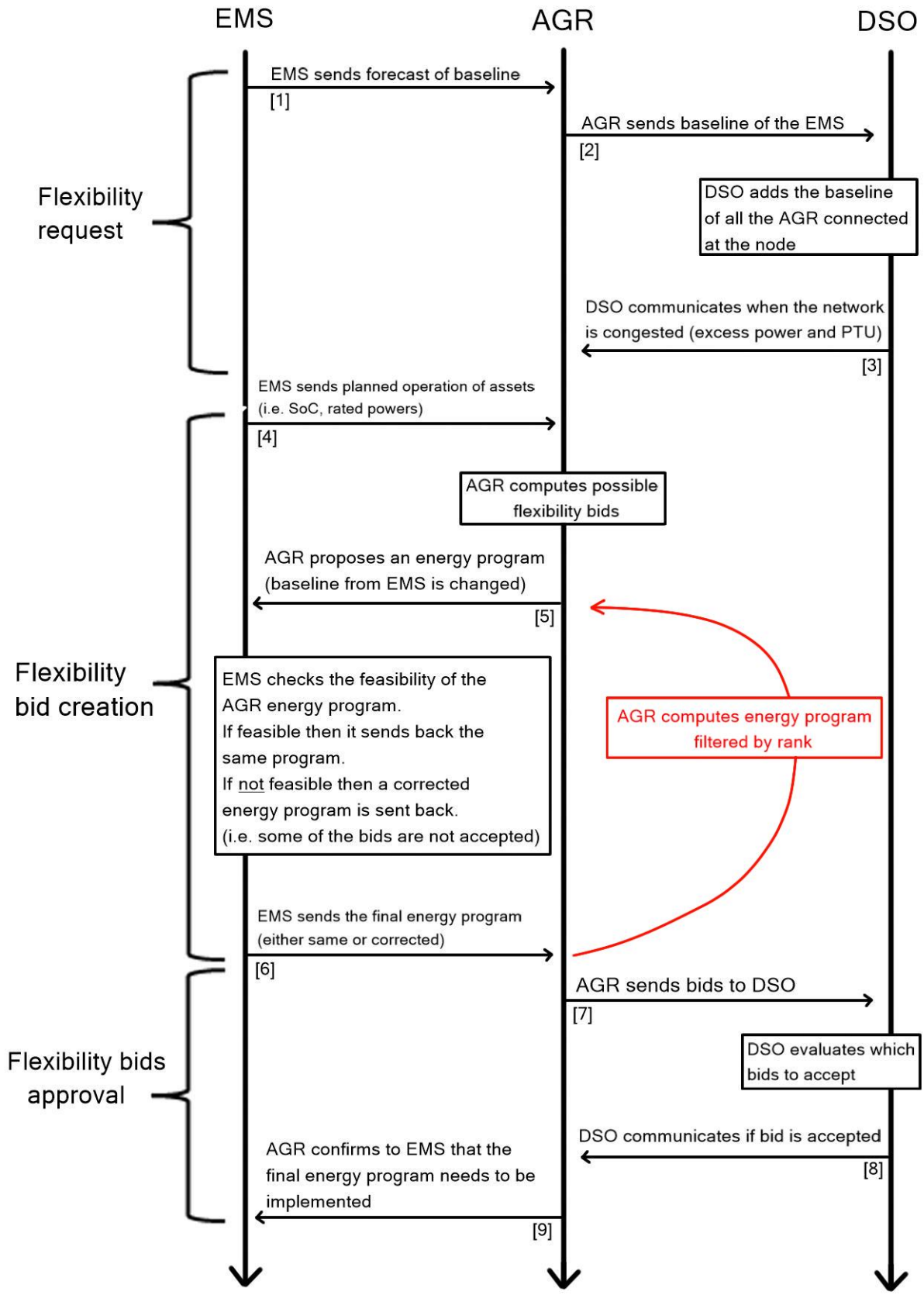


Figure 5.16: Schematic of the system interaction with rank filtering process.

5.3. Connecting all the actors together

This section looks at how each actor share the protocol messages with each other.

5.3.1. MQTT protocol

All the actors involved and the different interactions have been explained but it has not been described how they would exchange information. The EMS and AGR are two different entities that do not share the same code and thus the message defined above needs to reach one another. To do so, the MQTT messaging protocol has been chosen. The protocol is formed of two components: The MQTT clients and MQTT broker as shown in figure 5.17. The clients have to possibility to either publish or subscribe to a topic. When publishing, the client sends a message linked to a topic to the broker whose role is to redirect the message toward the correct client. The broker knows whose client is supposed to receive the message because said client is subscribed to a topic [28]. In this case, the client would be actors such as the EMS, AGR, and TGV. For example, the TGV sensor would send the data to the EMS, which will share it with the AGR. Once the AGR has computed the flexibility bids, it can share it back to the EMS. The advantage of using the MQTT protocol is that it is very versatile. At the moment there is only one client of each kind (i.e. one EMS, one TGV) but in the future, more actors may be involved (i.e. a digital twin). When said actors can be then easily connected to the system by simply publishing or subscribing to a topic, facilitating scalability.

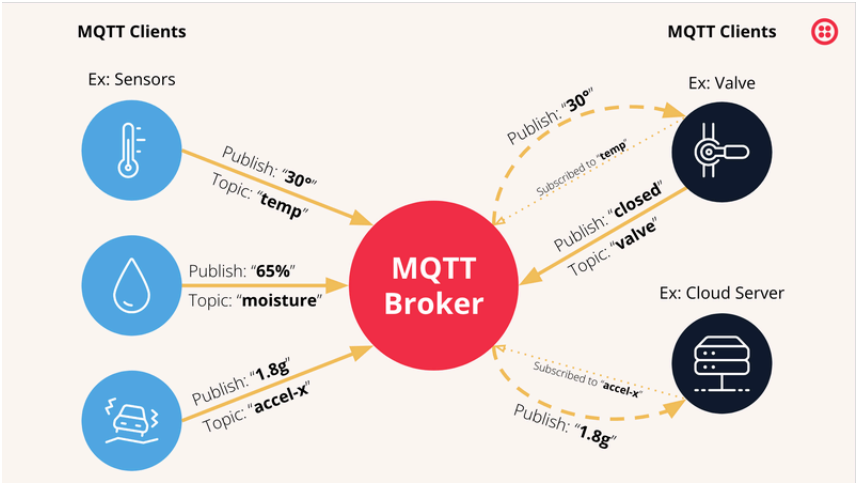


Figure 5.17: MQTT messaging protocol structure [28]

Figure 5.18 shows the different specifications of the MQTT messages. For each of them, the topic was defined as well as the rule (how often the message needs to be sent), the subscriber and sender, and last but not least the content of the message. It can be also noticed that there is an extra client included with the initial DT which stands for digital twin.

TOPIC	RULE	SUBSCRIBER	SENDER	CONTENT
Baseline	Every 15min	AGR, DT	EMS	Power exchanged with the grid and local consumption for every 15min of the next 24h.
Planned E program	Every 15min	AGR, DT	EMS	SoC (if applicable) and rated powers of the assets, local consumption.
AGR E program	Once before gate closure	EMS, DT	AGR	Energy program (modified baseline) + changes in planned operation.
Final Energy Program	Once after receiving the AGR E program	AGR, DT	EMS	Corrected energy program (modified baseline) + changes in planned operation.
AGR confirmation	Once after receiving the DSO bids response.	EMS, DT	AGR	Part of the Final E program and changes in planned operation that have been accepted by the DSO.

Figure 5.18: MQTT message specifications

5.3.2. OpenRemote

To make the connection between each MQTT client, the platform OpenRemote is used. OpenRemote is an open-source IoT platform that facilitates IoT device management [29]. Although OpenRemote offers many services such as asset management, location services, and data visualization, in this research it was mainly used as an MQTT broker. The reason why OpenRemote was chosen is because of its open-source characteristic. As explained, one of the requirements of the AGR program (but also the system) is to be open source to incentivize its usage.
At the moment of this study, the OpenRemote platform is locally hosted. The different actors involved were created as assets (i.e. the AGR and EMS), each composed of attributes that hold the messages defined in figure 5.18. Through a python script given in appendix B, it is possible to send JSON files over to OpenRemote which will distribute them to each of the assets. The platform also has the option of creating rules that enable certain behavior. For example, a rule has been created such that when the EMS asset receives a JSON file containing the planned operation, the message is then shared with the AGR assets. OpenRemote will then send the planned operation to the AGR program (or AGR python code) so that it can analyze the data to determine the available flexibility. Figure 5.19 shows the OpenRemote asset manager UI with the AGR asset opened and the different attributes with the rule state configuration enabled. This configuration is needed for the rules, shown in figure 5.20, to function.

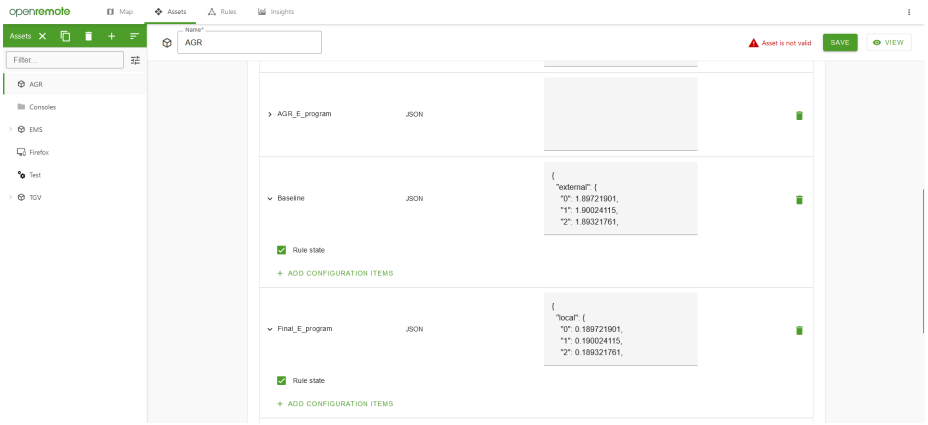


Figure 5.19: Example of OpenRemote asset with attributes and rule state configuration

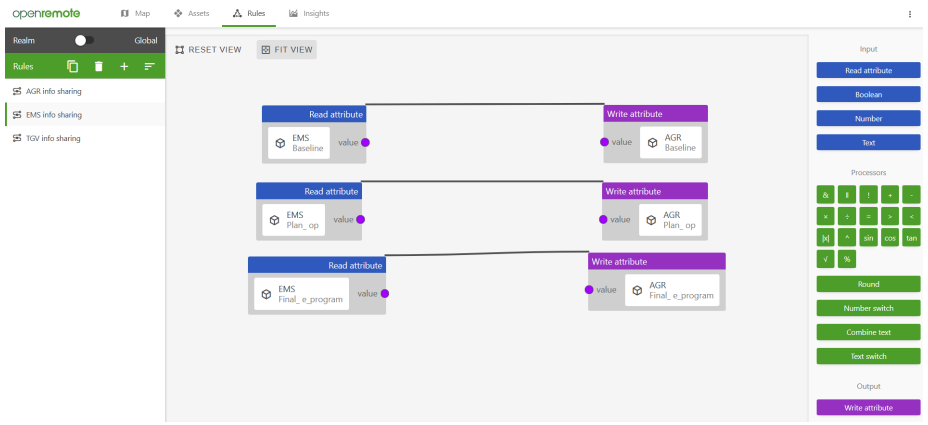


Figure 5.20: Example of OpenRemote rule for exchange of information within assets in OpenRemote

The OpenRemote operations are done through a separate Python script which is structured as such. First, the MQTT client configurations are set, these comprise the username, password, host, port, and clientID that have been defined in OpenRemote. These informations allow the Python code to connect to the OpenRemote session. Then it is the turn of the asset configuration which only includes the asset ID. With this ID, OpenRemote can know to whom the data is supposed to reach. Once this is done, the data that is sent to OpenRemote needs to be converted into JSON files. To do so, the data that is sent over needs to be first saved locally and then uploaded on the script. The data is then converted into a dictionary which then can be converted into a JSON file. Once this is done, the data can be sent over through OpenRemote. In the case that the python script needs to receive data then a function called JSON_2_CSV was created. When the actor subscribes to a topic, the message received is saved into a JSON file named data. This file is then converted to csv with the function mentioned and transported to the variables in python so that the code can do its computation. As each actor have different topics that that they publish to and subscribe to, a OpenRemote python script was created for both the EMS and AGR named OpenRemote_EMS and OpenRemote_AGR respectively. They are both identical, but the messages included that are they can publish and subscribe to have been tailored to figure 5.18. To then send the info, the script just needs to be called, similarly to the automated ranking process explained in section 5.2.6.

In this chapter the AGR program was studied in depth. The program's structure was introduced and then each section was individually discussed. It has been explained that the program was subdivided into four sections in order to satisfy the three structural requirements.

Interoperability, to have an AGR program compatible with as many EMS as possible. This requirement was satisfied by letting the AGR program handle as many tasks related to the flexibility bidding as possible. The flexibility request and offers are made through and only the AGR program. The AGR handles the determination of available flexibility, which creates flexibility bids and assigns a price for each bid. Nonetheless, the EMS is still needed in the process. The EMS must send a forecast of the baseline for the next 24h which can be challenging depending on the control methodology used (i.e. not all EMS may be able to forecast their baseline). The USEF Foundation has found a way to partially counter this by requesting only the baseline of the flexible assets, which makes the forecast easier as fewer loads need to be taken into consideration [4]. The EMS also needs to do a feasibility check which similarly to the baseline forecast can be difficult depending on the control method used by the EMS.
The second requirement was redundancy, to avoid any possible errors. The USEF Framework has already a set of safety nets implemented to avoid any communication errors and the proposed protocol uses a similar style. The changes in the baseline are checked and confirmed by sending over the same energy programs. This method has the goal of double-checking that the two actors are communicating (i.e. EMS and AGR) are agreeing on the same changes in operation. The protocol is also built so that the EMS has the final say which helps to avoid any possible misunderstandings. As eventually, the EMS is the one that needs to activate the flexibility, it is assumed that any program accept by it is feasible. If the EMS does not respond to the AGR Energy program proposal, then the AGR is not able to make any bids. Such a safety net makes sure that the AGR is unable to bid flexibility that was not double-checked by the EMS.
Finally, the plug-and-play was unlocked by clearly dividing the flexibility bidding into four sections each responsible for a different task. For each section, the message and data that are needed and generated were clearly defined so that changes and upgrades are clearly bounded. An example would be to implement a price forecasting algorithm to set a price for the flexibility bids, or creating a more complex optimization algorithm that can do other actions such as load shifting.

6

Validation in simulation and discussion

This chapter treats on the testing of the AGR program.
First the system and data available are introduced. Second, the Python code structure is explained. Finally, the results of the simulation are shown and discussed.

6.1. System and data available

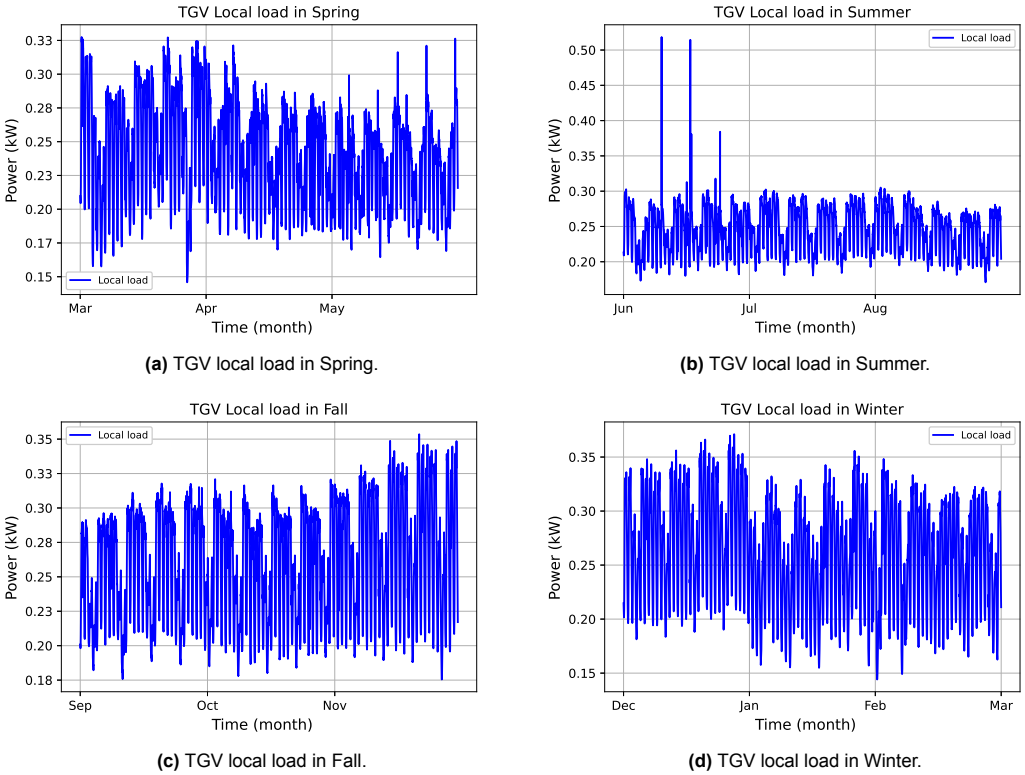
6.1.1. System

As discussed in the previous chapters the AGR program has been created with the intent of handling flexibility trades between the assets owners and the grid operator. TGV located on the campus of TU Delft perfectly fits these requirements as it is equipped with a diverse set of flexible assets that enables a wide range of flexibility possibilities. The concept is that the AGR program would be connected to TGV through OpenRemote, an online platform that would allow the AGR program and the EMS of TGV to share data with each other and/or any other additional actors (i.e. a digital twin).
The study on the AGR program was started after the involvement of the Intelligent Electrical Power Grids research group at TU Delft whose role is to create an EMS with different modes of operation. For each mode the EMS operates with a different goal namely, mode 1 for zero power exchange with the grid, mode 2 for net-zero exchange with the grid, mode 3 for reducing peak consumption, mode 4 to minimize the CO2 footprint of the grid intake, mode 5 for maximizing profit on the energy market and finally mode 6 for balancing service to grid operator and external actor. The EMS needs to be set in mode 6 so that the ARG program can communicate with it. At the time of writting this paper, the EMS does not yet have this mode installed which hinders the quality of simulation and testing of the AGR program. It is important that the reader understands the limit of the result.

6.1.2. Data available

In subsection 6.1.1, it has been explained that the system is yet to be operational so no data directly correlated to TGV exists so other data needed to be used. The data used is a package containing different types of timeseries of 15 minutes of various European countries. The one used are the load consumption and forecast in the Netherlands, the German solar generation and the day ahead arket

price of Austria from 2016 [30]. This data is run through the EMS program which outputs the EMS baseline and planned operation for the whole year. The EMS program creates a dataframe formed of the forecasted of the local load, of the SoC and power ratings of the storage assets, of the photovoltaic production as well as of the exchanges with the grid (baseline) for the whole year. With this data the AGR program was tested through simulations. As there is a lot of data, and each season has a different characteristics (i.e. greater consumption in winter), the simulations are divided into these four seasons each shown in figure 6.1. There are two points that need to be discussed when using this data.
Firstly, the winter season is composed of the month of January to March and then September to December. These months are stitched together as if they were one but that is not the reality. The EMS starts its simulation in January as if the system has just be put online. The problem with this is that the storage assets need to charge so the early winter months may not represent the SoC levels of the storage assets that the EMS would normally plan to have by that time.
For example the hydrogen storage is planned to be used during the cold months when there is not as much photovoltaic production [14]. In figure 6.2, the SoC of the storage assets throughout the year shows that the hydrogen is discharged during winter. However, the SoC is very low in January and might not represent correctly the start of the winter. For example, at the end of december the SoC is round 80% and it is likely that in January of the year after the SoC will not be around 30% as it is in this dataset. If so it would mean that over night the hydrogen storage losses 50% of its charge.
Secondly, it can be seen in figure 6.1 some high peaks appearing in end of Spring and Summer.
No explaination could be found for such peaks, an hypothesis is they are the result of some errors but it cannot be confirmed.



(a) TGV local load in Spring.

(b) TGV local load in Summer.

(c) TGV local load in Fall.

(d) TGV local load in Winter.

Figure 6.1: Local load for the different seasons.

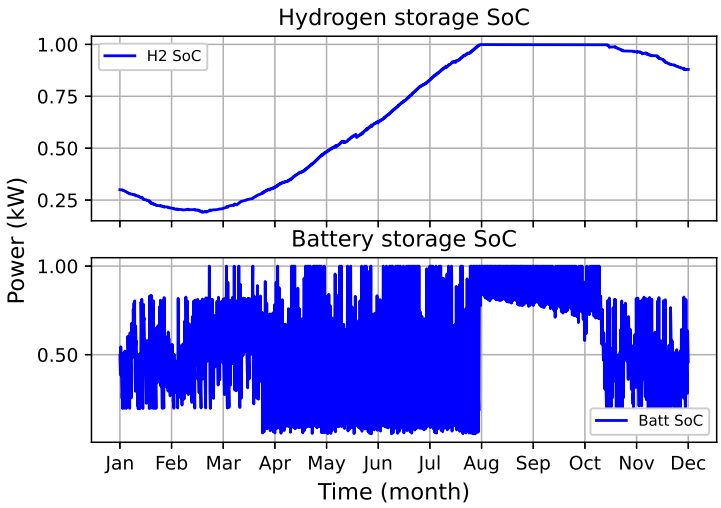


Figure 6.2: SoC of the storage assets throughout the year

Now that these details about the data is organised, the way that it is used needs to be addressed. The first topic to discuss is that the EMS only has mode 1 operational (no exchange with the grid), the baseline is then zero. This is a problem as the first step in the flexibility trading is adding up all the baseline together to see when the grid is congested. To simulate grid exchanges and possible congestion, it is assumed that there are ten other energy communities identical to TGV and all connected at the same node. Said energy communities have the same consumption pattern of TGV but do not have storage or generation assets and are thus powered by the grid. This leads to a node baseline shown in figure 6.3 which depicts the baseline at the node and the local load of TGV.,
It can be seen that the baseline at the node is the same of the TGV local load except ten times greater in magnitude. As explained in chapter 5 section 5.2.1 all power above the congestion limit is auctioned to the flexibility market where the AGR can bid.

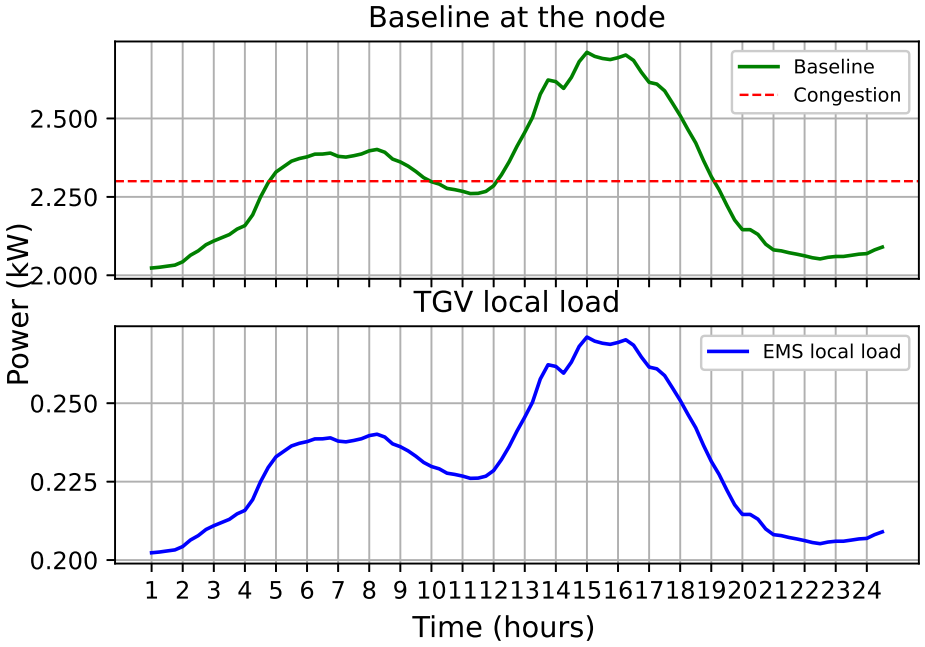
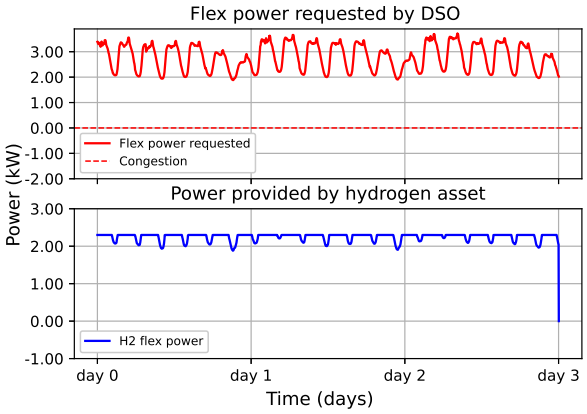
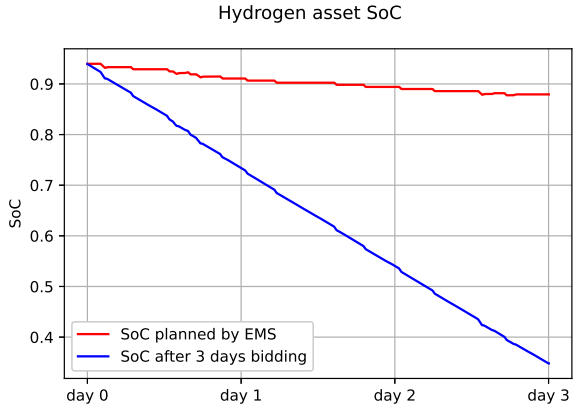


Figure 6.3: Baseline at the node and local load of TGV

Additionally, as the EMS cannot communicate back to the AGR which bids are feasible, it will be done manually to recreate the communication exchange. The bids that are most likely to be denied are in the morning or afternoon peaks (i.e. morning peak from 5am to 10am or 12pm to 19pm in figure 6.3). The train of thought is that during those periods TGV consumes the most which means that providing flexibility at those moments may be the most difficult as the system is self reliant and its storage powers all consumption. The EMS's refusal of the peak (either one or both) will be manually simulated (hardcoded). Furthermore the SoC is also used as an indicator for refusing a bid. For example, if the SoC reaches a low level then the bids are refused. The problem with this method is to define an appropriate SoC limit. For the case of the hydrogen, the storage capacity is quite large (the capacity is 90% of the yearly consumption of TGV), and because of its relatively low maximum rated power, the SoC does not rapidly decrease unless it is put into extreme situations.
The following simulation is done to display it. The congestion limit is set to 0 kW and only the hydrogen asset is available for flexibility bidding. Addtionally, the simuation is ran for 3 consecutive days during the period of 29th to 31st of december which is the moment when the baseline of TGV is the greatest. This equivalent to say that the hydrogen asset now has to power not only the local consumption that was initially planned by the EMS but also the node baseline that correspond to 10 times the EMS's planned consumption. The point of this test is to drain the hydrogen storage to its limit. Figure 6.4a shows the amont of power requested by the DSO and the power provided by the AGR. As it can be seen, the hydrogen asset provides power is at its maximum power rating for the majority of the three days. Small deviations are the result of the flexibility requested being less than the maximum power rating of the fuel cell or because the fuel cell is already in use and thus there is only that remaining amount of power left to be used to match the flexibility request. Figure 6.4b, shows the SoC after the three consecutive days. The red line is the SoC planned by the EMS or in other words how the SoC would have changed if not flexbility were to be traded. The blue line on the other hand is the SoC after the flexibility trading. As it can be seen, the SoC is around 30% which can be defined as a critical level, however, the point of this simulation is to show that the hydrogen asset must be put into intense conditions to reach such levels. Such situation is most likely not going to happen as the congestion limit is not at 0 kW and the EMS will never let the AGR match the flexibility request to its maximum for three consecutive days.



(a) The hydrogen asset providing power for three days when congestion limit is set a zero.



(b) The hydrogen asset SoC after three days when congestion limit is set a zero.

Figure 6.4: Hydrogen asset operated at intensive conditions.

As for the battery, the SoC varies a lot as it is used by the EMS to power day to day consumption and thus it is more likely that the introduction of flexibility trading will make the SoC drop further. A limit of 30% can be implemented such that if the SoC reaches said limit, all bids will be refused below this limit. This way the EMS'rejection is simulated in two ways. Manually, by removing peaks that are likely to be rejected and through the SoC.

The second topic is how the flexibility market was simulated. At the time of writting this paper there is no python code that can replicate the USEF market that is known of. The assumption is that TGV is the only active AGR in the node and thus he is the only bidder. It is a monopoly situation.
Accurate price forecast is quite important if other actors were involved in the flexibility bidding process. The AGR would need to offer a market competitive price if it wishes to win the bidding but in this case it is assumed that the DSO will accept whatever price the AGR offers. The price that the AGR will offer will be the day ahead electricity market price taken from the timeseries [30].

6.1.3. Python code structure

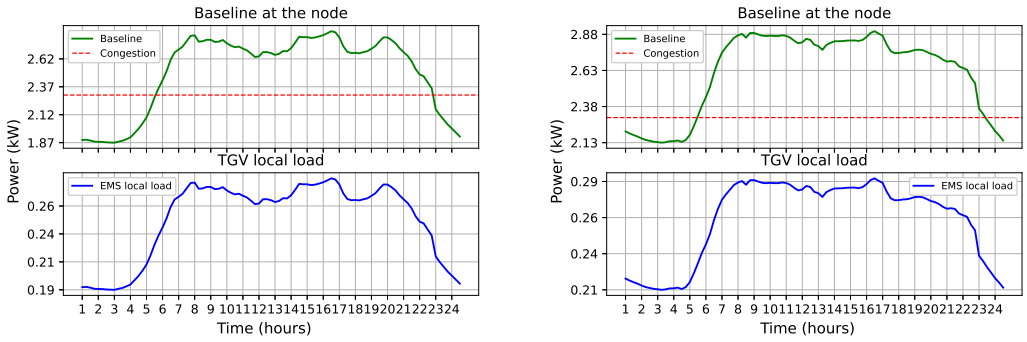
The python code is structured such that it follows the same step as described in chapter 5 although some of the interactions cannot be simulated as explained in section 6.1.2. Function have been created and called on the main.py script for easier reading. This subsection focuses on explaining the step by step process of the code to facilitate the understanding. Additionnally a list with all the function implemented is appended in the appendix B.
the first step is to forecast the possible congestion. This is done with the function DA_FR which takes in the baseline at the node (i.e. baseline computed by the EMS times ten) and the congestion limit. The function then returns all energy that is above the congestion as a dataframe collumn named FR_PTU which is the amount of power requested in flexibility for each PTU as well as the available power before the node baseline reaches the congestion limit.
With the known amount of energy requested, the optimization can be run to evaluate and create the flexibility bids. This is done using the function Flex_calc which takes as inputs, the EMS' planned operation and FR_PTU (flexibility requested). The optimization is done using Pyomo and an abstract model was chosen to be used. The reason why the abstract model was chosen over a concrete is because the AGR program should try to be as versatile as possible. Each system that the AGR program is partnered with may have different assets but the optimization is the same. With an abstract model, the AGR operator would need to load in the dictionary introduced in section 5.2.2 (or figure 5.9) and the optimization would be updated with the needed parameters (i.e. capacities and rated powers). At the moment the code can only accept updates for the battery and hydrogen assets as other parameters have not be introduced and modelled for the optimization. The output of the optimization is the change in SoC (SoC_change), the amount of power offered to flexibility (PshFlex), and the AGR energy program (AGR_E_program).
The amount of revenue can then be computed with the function Financial_calc and the bids can then be ranked using the function Ranking. Then the EMS will do a feasibility check of the AGR energy program, as explained in section 6.1.2, and communicate it back to the AGR as the Final Energy program (Final_E_program). Once the AGR receives the Final Energy program, it will run the script Ranking_Automated.py whose purpose is to check alternatives to the EMS's feasibility check energy program as explained in section 5.2.6. The script is mainly composed of two function; Difference_calc, that computes how much flexibility power is provided by the rank-filtered energy program compared to the AGR energy program and Rank_analysis, that calculates the revenue of the rank filtered energy program. The output of the Ranking_Automated.py script are two dataframes. The first one is named Analysis_opt which gives the following information: the total amount of power provided and the revenue of the rank filtered and the EMS feasibility checked energy programs but also the revenue and power served ratios of the two programs and the optimal rank computed. The second dataframe is Analysis, and it contains the same information except for each rank up to the optimal rank.

6.2. Simulation result

6.2.1. Flexbility Trading over one day

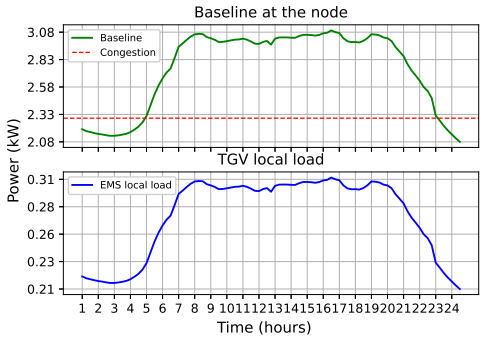
In this simulations the congestion limit is set at a 2.30 kW limit. For each season, only one day was chosen. The choice was made such that a good portion of the baseline is above the congestion limit so that the AGR program can do bids and the storage assets' limits are put to the test.

First, the baseline at the node and local load of TGV is examined. Figure 6.5 depicts it for the four seasons. Although these are singles days, the different characteristics of the four seasons are quite clear. The hotter months, Spring and Summer, have their local load consumption of smaller magnitude compared to the colder seasons of Fall and Winter. In fact if the average consumption for season of this dataset is taken, it yields to an average of 0.239 kW in Spring, 0.244 kW in Summer, 0.257 kW in Fall and 0.259 kW in Winter. This is quite relevant as the congestion limit has been set to 0.23 kW meaning that the colder seasons will request more flexibility than the hotter one.

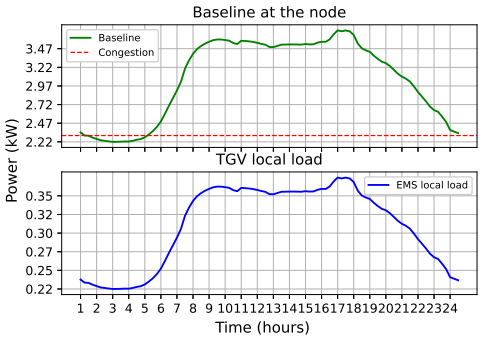


(a) Baseline at node and TGV local load in Spring.

(b) Baseline at node and TGV local load in Summer.



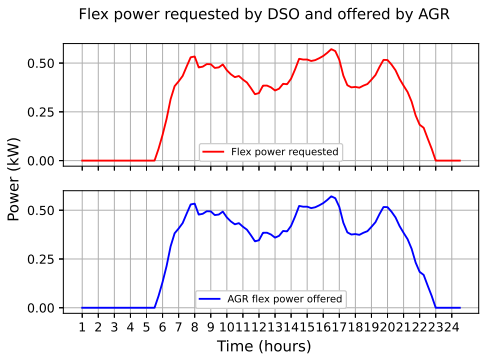
(c) Baseline at node and TGV local load in Fall.



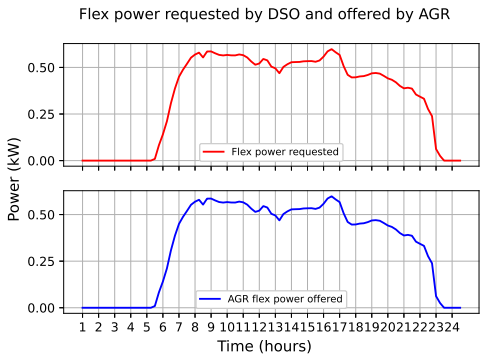
(d) Baseline at node and TGV local load in Winter.

Figure 6.5: Baseline at node and TGV local load for different seasons.

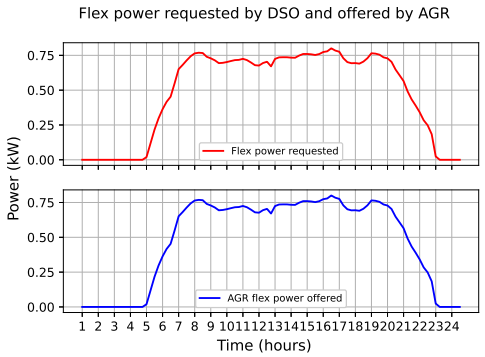
Figure 6.6 shows the flexibility requested by the DSO (red trace) and the flexibility offered by the AGR (blue trace). As explained in figure 6.5, the consumption is greater during colder season and thus a greater amount of flexibility is requested during that time period. In fact this can be seen in this snapshots where in the hotter months the power requested is on average around 0.50 kW or lower while for the colder season the power requested is around 0.75 kW and 1.25 kW for Fall and Winter. For all of the four seasons the AGR has no problem supplying the needed flexibility as the blue trace is the exact same of the red trace. The area under the blue trace is the amount of energy that is reduced from the node baseline. That energy is now powered by TGV's storage and thus the local load will increase.



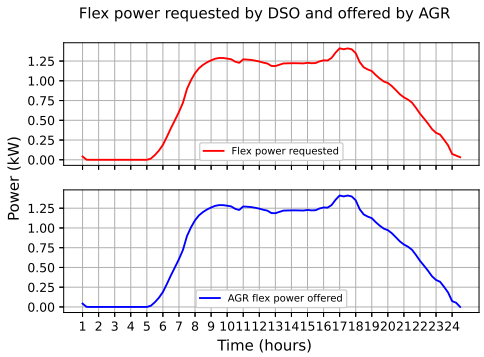
(a) Flexibility requested and offered by the AGR in Spring.



(b) Flexibility requested and offered by the AGR in Summer.



(c) Flexibility requested and offered by the AGR in Fall.

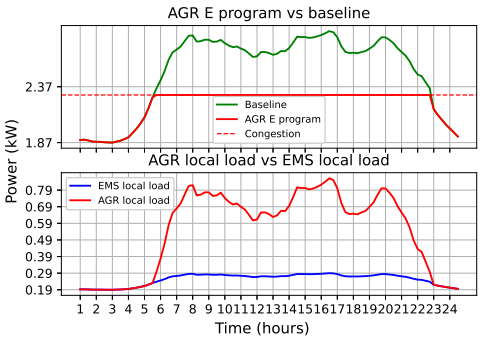


(d) Flexibility requested and offered by the AGR in Winter.

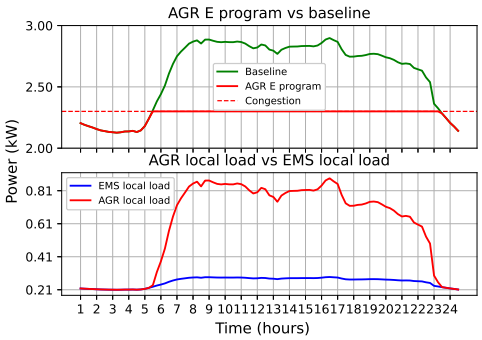
Figure 6.6: Flexibility requested and offered by the AGR for different seasons.

Figure 6.7 shows the energy program and local load proposed by the AGR (red trace) compared to the baseline and local load that was computed by the EMS (green and blue trace respectively).
When looking at the energy program it can be seen that, once the exchanges with the grid reaches the congestion limit, the AGR energy program flattens out and stays at that level. This is because all load is now being powered by the storage unit and not the grid anymore. In fact in the AGR versus EMS local load plot, the increase in consumption is the result of the flexibility bids and as expected, the local load experiences a greater increase in the colder seasons compared to the hotter seasons.
The AGR energy program can be sent to the EMS which will verify the feasibility of the bids and communicate back to the AGR what bids it is willing to accept. The EMS does not need to know the amount of power or change in SoC of each asset but only the AGR energy program (change in baseline) as it will evaluate how close it can match it.

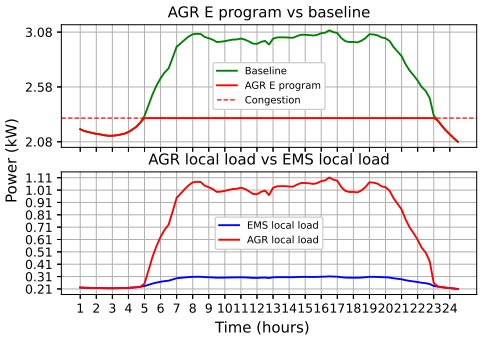
In this first subsection, the flexibility trading over one day for four different seasons has been shown. While one day is just a snipplet and does not represent the whole season, the days were chosen such to best showcase the characteristics of each. As it could already be seen in figure 6.1, the colder seasons result in greater consumption compared to the hotter seasons. This means that more flexibility is requested requiring more energy from the AGR. It has been shown that for the each day of the season the AGR did not have a problem with supplying flexibility and an AGR energy program was created. Said program is proposed to the EMS which will assess its feasibility. Subsection 6.2.4, looks into a rejection from the EMS and what can the AGR do in such case. It is important to specify that the reason why the simulation was ran only for one day is because it is very unlikely that the AGR will be allowed by the EMS to fully match the flexibility request for multiple consecutive days as it would just drain the storage assets. The simulation over one day is supposed to demonstrate the AGR's working in different situations.



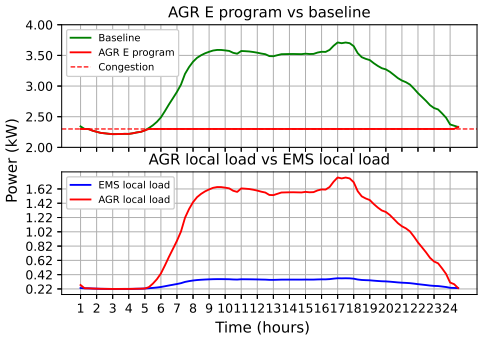
(a) AGR and EMS energy program and local load in Spring.



(b) AGR and EMS energy program and local load in Summer.



(c) AGR and EMS energy program and local load in Fall.



(d) AGR and EMS energy program and local load in Winter.

Figure 6.7: AGR and EMS energy program and local load for different seasons.

6.2.2. Sensitivity analysis of flexibility trading over a day

A sensitivity analysis is carried out to test the limits of AGR trading. The goal of this analysis is to see what conditions are needed for the AGR to not be able to fully provide the flexibility requested. One situation can be that there is not enough charge in the storage assets (i.e. the storage is fully discharged). In the section 6.1.2 and figure 6.4b it has been shown how unlikely it is for the hydrogen storage to fully discharge. It is more likely to happen for the battery as the SoC varies greatly as shown in figure 6.2. Contrary to the hydrogen asset that has its charging and discharging decoupled, the battery discharging is halted when the asset is being recharged. This is a limiting factor if the hydrogen asset alone cannot provide enough power to fully supply the flexibility demand and needs the 'help' of the battery asset.
Figure 6.8 shows how the AGR is unable to provide all the flexibility requested. The moment that the battery goes into charging mode, highlighted by a blue dotted vertical line, the flexibility provided becomes constant at 2.3 kW, the maximum rated power of the hydrogen asset. It is only after the battery is no longer charged that the AGR can once again match the flexibility requested.

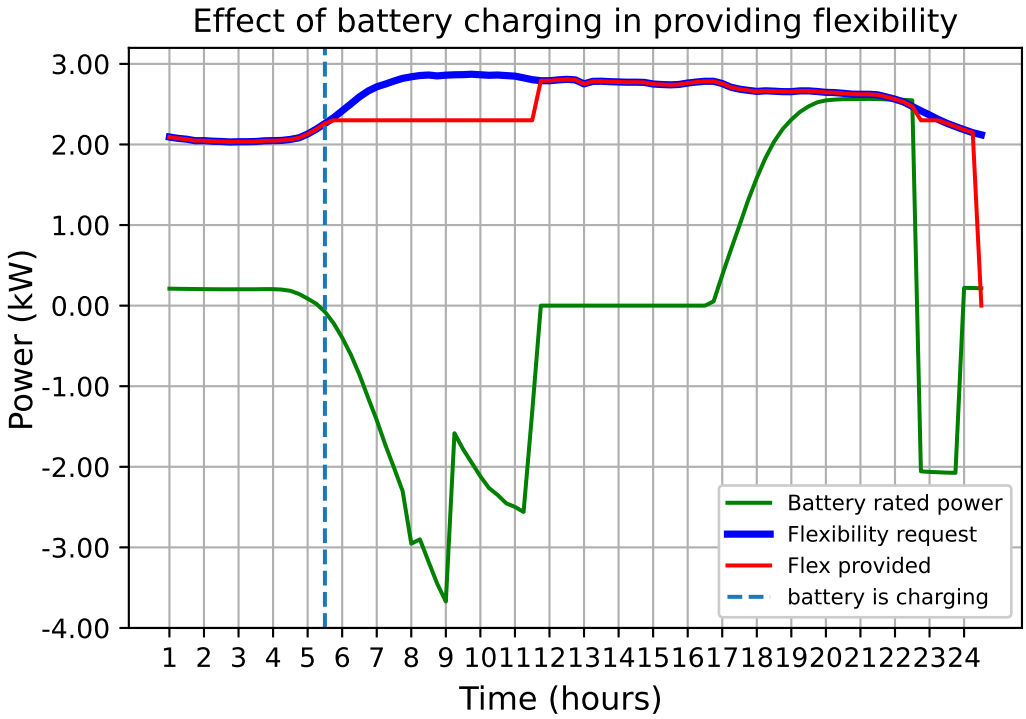
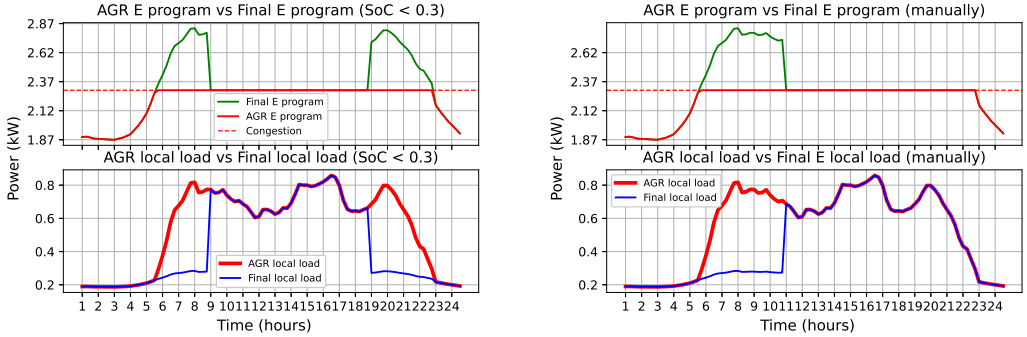


Figure 6.8: Battery charging effect on flexiility provision

6.2.3. EMS feasibility checks

Once the AGR has send its energy program to the EMS, this latter needs to check the feasibility. As explained in section 6.1.2, the EMS' decision is simulated by rejecting manually selected sections of the AGR bis and as well automatically rejecting bids when the storage asset is below 30%. Figure 6.9, shows the two methods described. On the left in figure 6.9a, the EMS refuses the morning from 5 AM to 9 AM and the evening bids from 19 PM to 23 PM due to the SoC being too low. Figure 6.9b on the other hand has the morning peak from 9 AM to 11 AM rejected because that time period was manually selected.



(a) Simulation of EMS feasibility check using SoC as an indicator. **(b)** Simulation of EMS feasibility check performed manually.

Figure 6.9: The two methods to simulate EMS's feasibility check in Spring

An extra simulation is done to show the effect of the SoC limit rule. The system is put under extreme conditions similar to figure 6.4, with the congestion limit set to zero. This time however, the battery asset can also be used by the AGR for the flexibility bids. The hydrogen asset alone cannot match the flexibility request and the battery needs to be used. A section of the data where the battery asset had a low SoC was also chosen for this simulation.
Figure 6.10 shows how the SoC limit rule comes into play. The figure depicts the AGR proposition (red trace) which is the changes in the energy program and local load after flexibility bids, the EMS feasibility check (green trace) which is the response of the EMS to the AGR's proposition and the battery's SoC (blue trace). As shown, when the SoC reaches 30% the EMS refuses all bids until the SoC is back above the limit. This happens around 5 AM to 12 PM AM and as well at 17:30 PM to 22 PM. From 9:45 AM to 15 PM the AGR is unable to match flexibility bids because the battery asset is being charged making it unusable and the flexibility power requested is too much for the hydrogen asset to match it on its own.

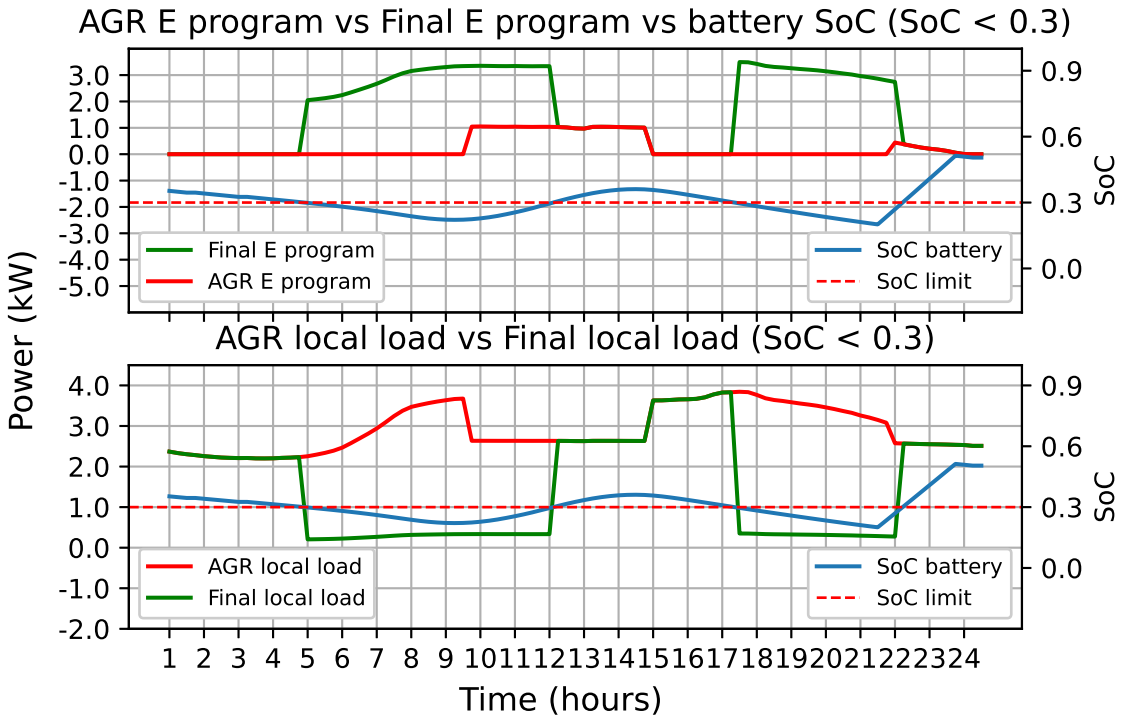


Figure 6.10: EMS feasibility check using SoC as an indicator.

6.2.4. Ranking and Ratio

Ranking to maximize revenue

In chapter 5 section 5.2.3, the Rank metric was introduced. As explained, each bid is ranked based on the revenue. The greater the rank, the more revenue the flexibility bid brings in compared to the maximum revenue possible for that time horizon (in this case one day). Figure 6.11 showcases the revenue ranking of all bids for the day in spring shown previously in figure 6.5a. The blue trace is the power offered as flexibility bids by the AGR while the red bar plots are the ranking on each bid. Only the bids happening at the exact hour are shown but in reality for every 15 minutes bids the rank is determined and thus in between hours the ranks can vary. This is intendented to help the reader to understand the graph with more ease.

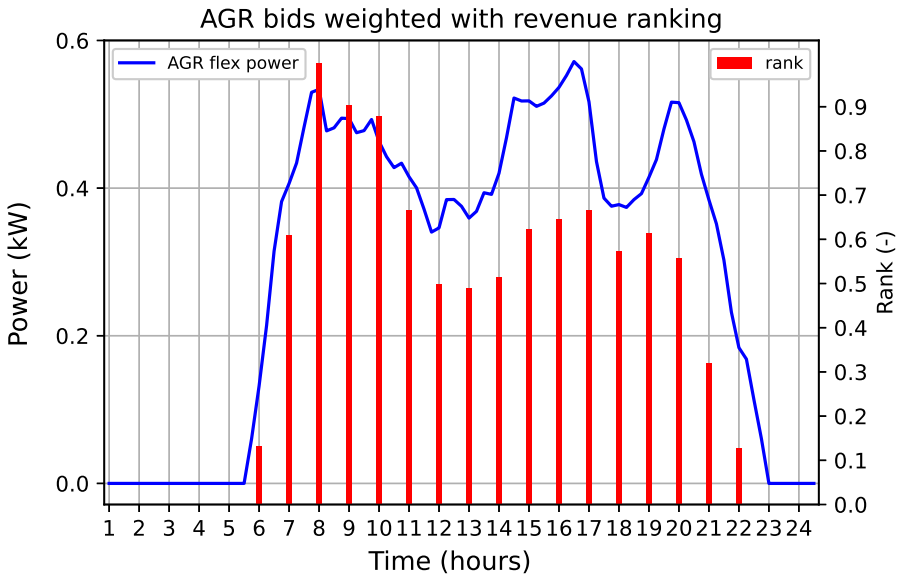


Figure 6.11: Bids ranks for a day in Spring

To show the concept of rank, the situation shown in figure 6.9b is used. The EMS has done its feasibility check and has decided that from 5:30 AM to 11 AM the AGR's bids are not accepted.
Table 6.1, lists the amount of energy and revenue for different cases. The AGR E program is the energy program proposed by the AGR, while the Final E program is the energy program approved by the EMS after a feasibility check. As it can be seen after the EMS's correction the AGR is allowed to provide 19.52 kWh compared to 28.18 kWh, that is a 30.73% decrease in energy served leading to revenue drop of 36.42%.

Case	Energy served (kWh)	Revenue (€)
AGR E program	28.18	865.14
Final E program	19.52	550
E program ranked (>0.6)	17.27	581.7
E program ranked (>0.5)	22.94	749.08
E program ranked (>0.7)	8.71	332.45
E program ranked (>0.57)	18.85	631.63

Table 6.1: Energy served and revenue for different cases.

This is where the revenue rank comes into play. As explained in section 5.2.3, the rank serves as filtering method to choose the bids that bring most revenue and discard the ones that are worth less. The EMS feasibility check refused the early bids which, as seen in figure 6.11, have a high rank and thus lead to a high revenue loss.

A test case is done where all bids above 0.6 are selected while the rest is ignored. The resulting energy program and local load is shown in figure 6.12. The AGR energy program and local load fluctuate as the bids in very early morning, midday and afternoon have a rank below 0.6.
In table 6.1 it can be seen that filtering of bids through ranks lead to total energy served of 17.27 kWh which is 11.52% less energy than what the EMS initally wanted to provide after its feasibility correction. It can also be seen that the total revenue is 581.7 €. This is 87.7 € or 14.85% more revenue than what the AGR would get if it follows the EMS's feasibility correction. Thus through filtering it is possible to find a compromise between the EMS preference, to be energy-conservative mindset as explained in section 3.2.2, and the AGR preference, maximize economical yield as explained in section 3.2.3. The amount of energy provided is less than the EMS correction but the revenue is greater.

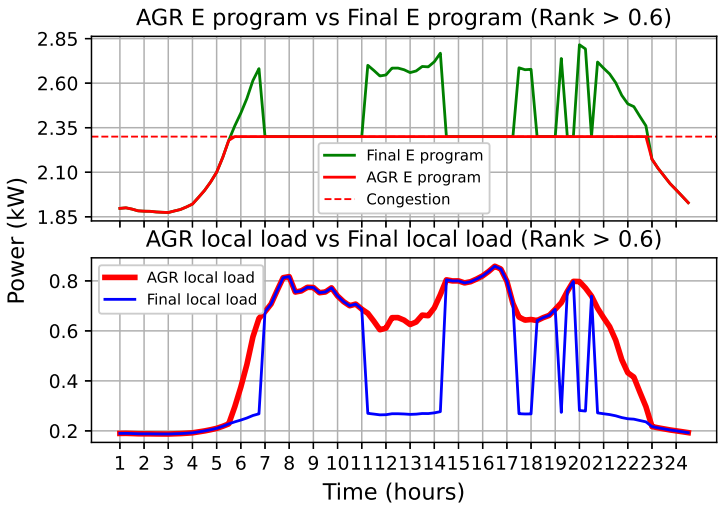


Figure 6.12: AGR Energy program and local load with big ranking filtering (>0.6)

As explained in section 5.2.6, chosing the correct rank is important. For example, a case study where only bids ranking above 0.7 are used. Figure 6.13 shows how the energy program and local load would look like. Fewer bids are proposed leaving a large portion of the midday to early evening without any bids offered. Table 6.1, shows that while the amount of energy served is less (pleasing the EMS preferences) the revenue is smaller than the Final energy program proposed by the EMS. It makes no sense for the AGR to propose such an energy program as it would yield a smaller revenue compared to the EMS's correction.

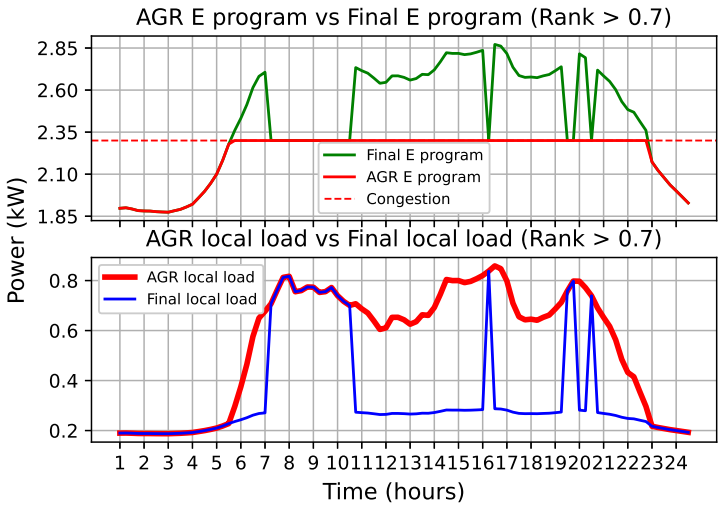


Figure 6.13: AGR Energy program and local load with big ranking filtering (>0.7)

Likewise if the rank filtering is done at a too low rank it is likely that the EMS will not accept it. Figure 6.14, shows the energy program and local load when bids are filtered for ranks above 0.5. In table 6.1, it can be seen that such filtering yields a larger revenue compared to the EMS's correction. However, the amount of energy served is also greater than the feasibility correction which likely will be denied by the EMS.

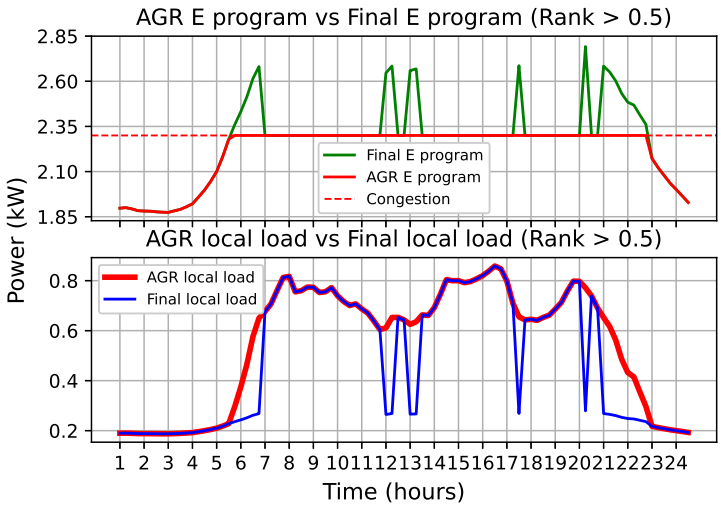
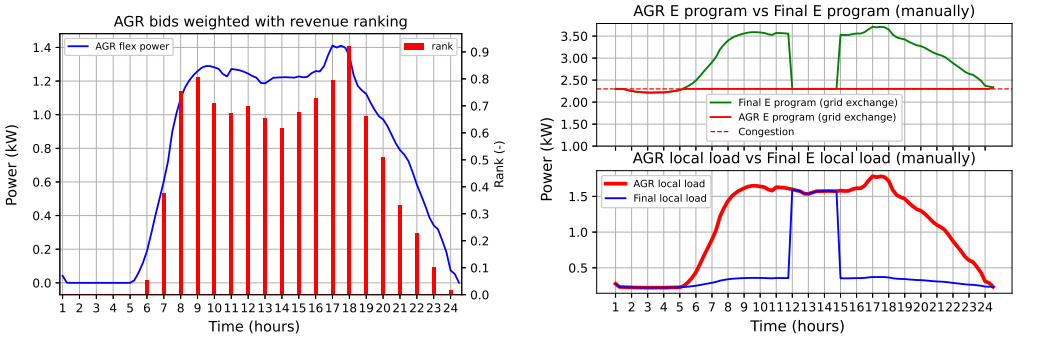


Figure 6.14: AGR Energy program and local load with big ranking filtering (>0.5)

6.2.5. Sensitivity analysis of Ranking

A sensitivity analysis is carried out to see the limitation of the ranking system. In the first analysis, the ranking process is tested for cases where the EMS decides to reject bids for both morning and afternoon peaks. This test case denies a lot of flexibility bids from the AGR and thus it is interesting to see how the ranking process performs under such circumstances. For such test case, the winter season is used because the bid rank profile during the peak hours, shown in figure 6.15a, are greater compared to the bid ranks shown in figure 6.11 that has bids with smaller ranks in the evening peak. Figure 6.15b, shows what the energy program and local load look when the morning and evening peaks are rejected. As it can be seen, only during the midday the EMS accepts the AGR bids.



(a) Bids rank for a day in winter

(b) Simulation of EMS feasibility check performed manually

Figure 6.15: other test case for morning and afternoon bid rank test

Different ranks are tested and it is found that there is no rank for which filtering the bids can lead to greater or equal revenue and lower or equal amount of energy served compared to the EMS's feasibility check. In this case, the AGR best offer is the initial Final Energy program of the EMS after the feasibility check. This is an example of the limitation of rank filtering.

A sensitivity analysis is also done with regard to rank incrementation. As explained in 5.2.6, the step that the rank is incremented at every iteration of the automated script can affect the result. The greater the rank, the more suboptimal the result. In the previous examples, the rank only increases by 0.1. The script is run with the same data but this time the rank is incremented by 0.01. Table 6.1, shows that the optimal rank is then 0.57 compared to the previously found 0.6. With a revenue of 631.63 € , 7.9% greater than the previously found optimal rank, and a total flexibility served of 18.85 kWh. A setback with a smaller rank increment is that it takes more time for the script to determine it. While this is not a huge setback when communicating with one EMS, the latency increase may be perceived when more EMS are involved and the AGR needs to find an optimal rank for each of them.

The price forecast variations are also tested. As explained in section 5.2.3, price forecasting can affect the efficiency of flexibility bidding. Random noise is introduced in the price by using a normal distribution function. Different deviations are used to simulate possible fluctuations. Three deviations are used: 10% of the maximum value of the day ahead price, 10% of the mean, and 10% of the minimum value, all with a mean of zero. The idea is to expose the price to three different forecasting errors. The three different price deviations are showcased in figure 6.16. For each case, the Ranking_automated.py script is run again to find the new optimal rank, power served, and revenue for that optimal rank. Then the script is run again but this time using the rank that was originally computed using the original price forecast. As the test is performed using the same price of section 6.2.4, the original optimal rank is 0.57. The goal is to compare the performance of the original optimal rank versus the real one. The results are presented in table 6.2 and are organized as such. First, the revenue and power served for the actual optimal rank are listed (optimal) and then the revenue and power served for the originally computed rank of 0.57 (suboptimal). As it can be expected, the greater the price deviation, the greater the revenue difference. What can be noted is that as the deviation size increases, the rank decreases. A reason can be that many bids have a smaller price than forecasted and thus more energy needs to be provided. The EMS preference is still respected, however, in the case of large and medium deviation, the AGR would lose a significant portion of its revenue because the rank would be too high (16% and 34% decrease respectively). Only the small deviation did not change and the rank and thus the AGR original rank forecast would be correct.
More simulations can be run to get a more in-depth understanding of how price deviation can affect the ranking system. Nonetheless, this highlights how an accurate price forecasting algorithm is essential to the correct operation of this AGR program. The appendix A shows the bid rank and AGR program for each case.

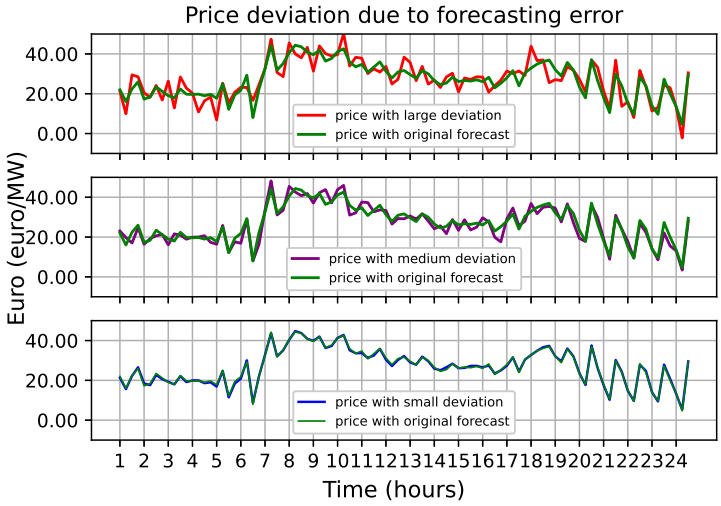


Figure 6.16: Three types of price deviations due to forecasting error.

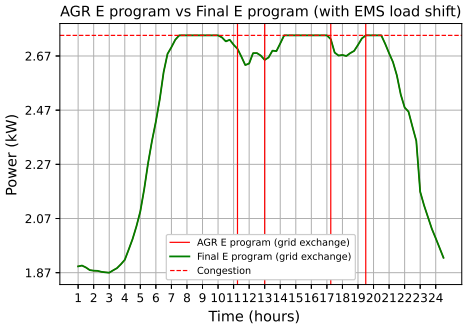
Case	Energy served optimal (kWh)	Revenue optimal (€)	Energy served suboptimal (kWh)	Revenue suboptimal(€)
Large deviation (>0.48)	18.99	629.35	15.29	539.17
Medium deviation (>0.47)	19.17	649.15	13.24	482.91
Small deviation (>0.57)	18.88	631.56	18.88	631.56

Table 6.2: Energy served and revenue for different price deviation cases.

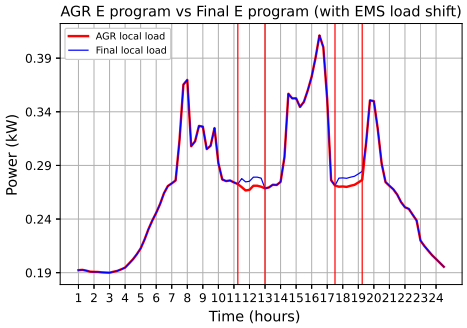
6.3. Future Implementation of the AGR program

As previously mentioned, the AGR program still needs to be tested with a real EMS to comprehend the pros and cons of the protocol and program. While a lot of errors cannot be foreseen, the purpose of this section is to highlight the possible adversities that will rise when connecting the two actors together.

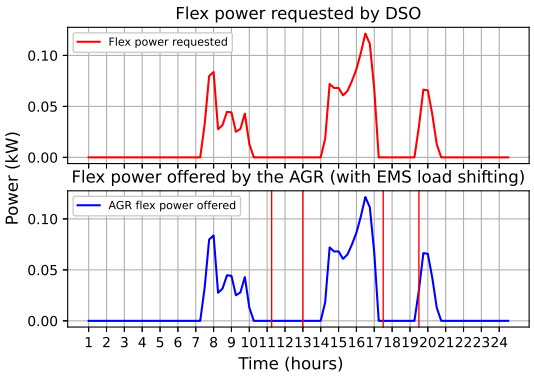
The one that will affect the AGR program working the most is the EMS feasibility check. At the moment it is simulated as a yes or no situation where the EMS rejects a part and accepts the rest untouched. In reality, when the EMS does the feasibility check, it will not only reject some parts but also create some deviations in the program. The reason for such deviation comes from how flexibility is provided.
In section 5.2.1 it was explained that the AGR program build can only do load shedding (or peak shaving). While, indeed the AGR program is able to evaluate and create flexibility bids that can only do load shedding, the EMS may not want to implement it that way. As the storage gets discharged, the EMS may need to shift some of its production to another time of the day.
Figure 6.17 simulates this effect. For the EMS to match the AGR energy program, its planned operation changes in an increase in local load from 11 AM to 13 PM and from 17:30 PM to 19 PM delimited by the red vertical lines shown in figure 6.17b. This is equivalent to a load shift.
As it can also be seen, the deviations only happen for the local load and not the energy program or the provided power. This means that the EMS is able to match the AGR program by adapting its local storage usage and consumption at a later time without exchanging power with the grid and effectively not affecting the flexibility bids.



(a) EMS's feasibility check deviation in grid exchange



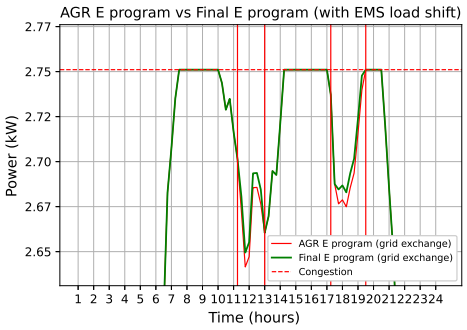
(b) EMS's feasibility deviation in local load



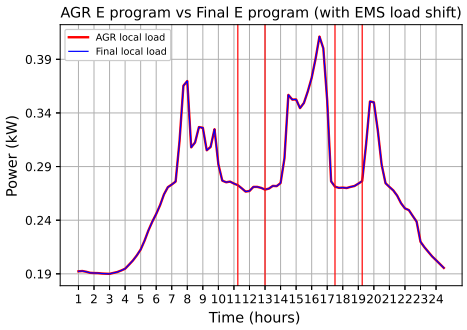
(c) EMS's feasibility deviation in power provided

Figure 6.17: EMS's feasibility deviation when power is locally provided

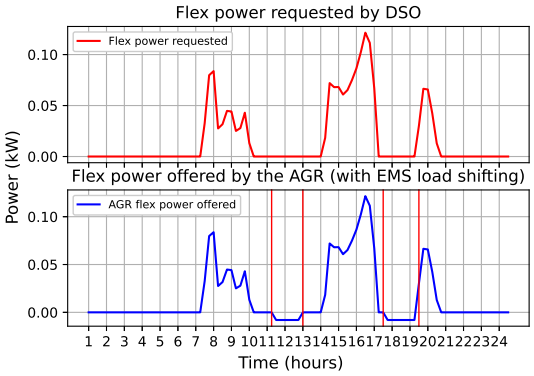
However, the EMS may decide to match the AGR energy program by exchanging energy with the grid. In this case, the load shift is not affecting the local load but rather the energy program and power provided. This effect can be seen in figure 6.18a with the Final Energy program having the increase in grid consumption during the period delimited in the red vertical lines. Likewise, the flexibility energy provided, shown in figure 6.18c, now has negative values as now the EMS offers to consume energy during those periods to match the flexibility requested by the DSO.



(a) EMS's feasibility check deviation in grid exchange



(b) EMS's feasibility deviation in local load



(c) EMS's feasibility deviation in power provided

Figure 6.18: EMS's feasibility deviation when power is grid provided

These deviations are no problem for the AGR program until they do not create other congestion. Fortunately, as explained in section 5.2.1, the USEF protocol is created such that the network's available energy is given. By knowing the margin of available energy the AGR can check whether the deviations are still within the margins of available energy and thus do not create more congestion. These deviations need to be communicated to the DSO as the energy program that was initially submitted (baseline) is now different. Indirectly the AGR is now bidding also for load shifting through the EMS. The following example explains this. A deviation due to the EMS feasibility check results in an increase in consumption during the middle of the day. At that time, there is still available energy before the grid is congested and thus the deviation creates no problem. However, when the AGR sends its bids to the DSO through the USEF protocol, it has to include the fact that there will be an increase in consumption. From the DSO perspective, it sees the AGR flexibility bids as a decrease in consumption resulting in an increase in consumption at a later moment of the day. Effectively, this is the equivalent of a load shift although the increase may not be of the same quantity as the energy shed as part of it can be supplied by storage (so a combination of storage and load shifting).
As mentioned, this is not a problem until the deviation does not create another congestion. For it to happen, there are two conditions that need to be fulfilled. The first one is that the EMS decides that the load shift will be powered using the grid, and the second is that the magnitude of the load shift is large enough to affect it. In this scenario there are no other active participants in the grid that can solve the congestion, meaning that if TGV does not solve it then no one will. In the case other participants are active then the deviation simply means that the AGR of TGV bids less flexibility. Rank filtering can also help to avoid more congestion due to deviations. As explained, the bids get filtered with the ranks and less energy needs to be provided, theoretically making it easier for the EMS to match the AGR energy program and requiring deviations that are less likely to create new congestion. More testing is needed to see how it would affect it.

6.4. Testing with a real EMS

As explained in section 6.1.2, the EMS is not fully operational at the time of writing this report. A first instance of the EMS program was created to test how the AGR program and EMS would perform. This section looks at how the EMS tries to match the operational changes proposed by the AGR. The data used is the four seasons sampled in section 6.1.2 and used for the first simulation validation in section 6.2.1. First, an in-depth look is given on how the EMS matches the winter season flexibility request and then the other seasons are analyzed.

6.4.1. EMS operation for winter

The EMS has different ways to provide flexibility and in the case of the winter season, all the different ways are employed which unveils both the range of possibilities but also limitations. Figure 6.19 shows the local load of the AGR and the EMS after the feasibility check. As it can be seen, the EMS does not manage to fully match it, and a small margin between the AGR local load and the EMS's remains. To understand why this is the case, the way that the EMS decides to provide flexibility needs to be looked into more detail.

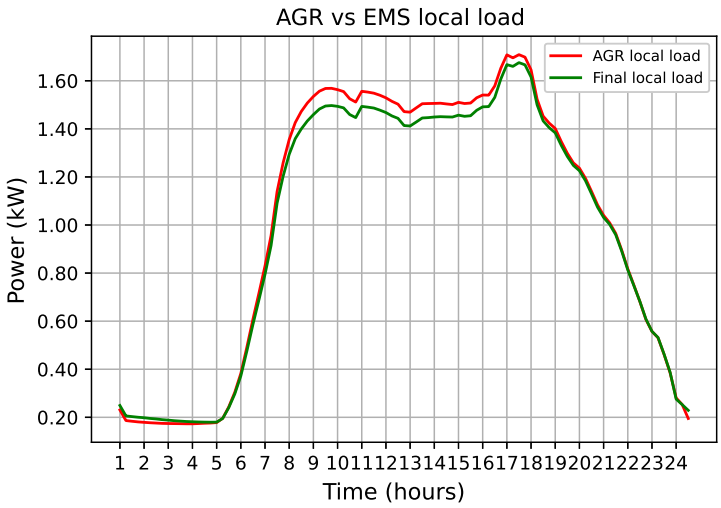


Figure 6.19: Local load after EMS's feasibility check in Winter

In figure 6.20, the different ways that the EMS provides flexibility are shown. It can be seen that the battery is being utilized from the start of the day to 10 AM together with the help of the solar from 8 AM. It is then the combination of PV production and hydrogen until 14 PM and then again PV production and battery. At 16:30, the EMS uses the grid to match the flexibility request. Two things need to be discussed. Between 10 AM and 14 PM, the battery is under charge and the EMS uses the PV production to charge it. The PV production is much more than what the battery needs and the EMS uses the remaining power in combination with the hydrogen to match the flexibility demand. The other detail is that from 21 PM the battery is again under charge and the hydrogen is discharged. While no definite reason can be given for this behavior, it is hypothesized that the hydrogen asset is used to charge the battery. Please take into consideration, this is a supposition.

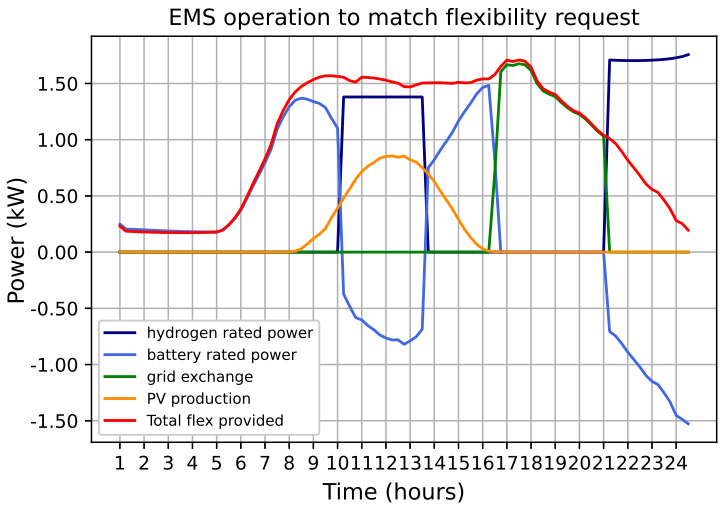


Figure 6.20: EMS's way to match the flexibility request in Winter

Because the EMS uses the grid to match the flexibility request, the bids of the AGR will change and needs to be communicated to the DSO as explained in section 6.3. Figure 6.21 shows how the flexibility offer of the AGR will look like. As it can be seen from 16:30 PM to 21 PM, the power is negative as the AGR would take energy from the grid.

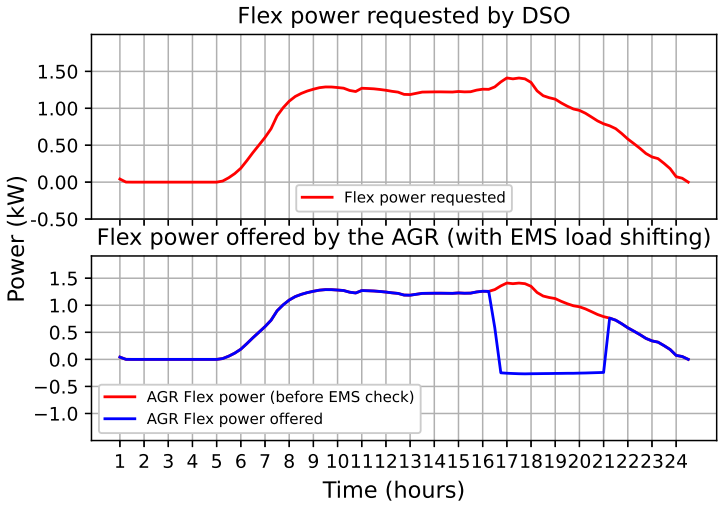


Figure 6.21: Flexibility bids before and after the feasibility check in Winter

As explained in section 6.3, the deviations are not a major problem until they do not create new congestion. This does not happen when the increase in grid exchange happens if there is available energy as explained in section 5.2 and shown on figure 5.4. This is not the case for this simulation. The EMS fully matches the flexibility request of the DSO by using the grid which is not advantageous as the exchanges with the grid need to be reduced. Figure 6.22 shows how the energy program changes with respect to the AGR energy program and the initial node baseline. As it can be seen there is a spike at 16 PM when the EMS starts using the grid to avoid congestion. Not only is the congestion not solved but even more energy is being exchanged with the grid.

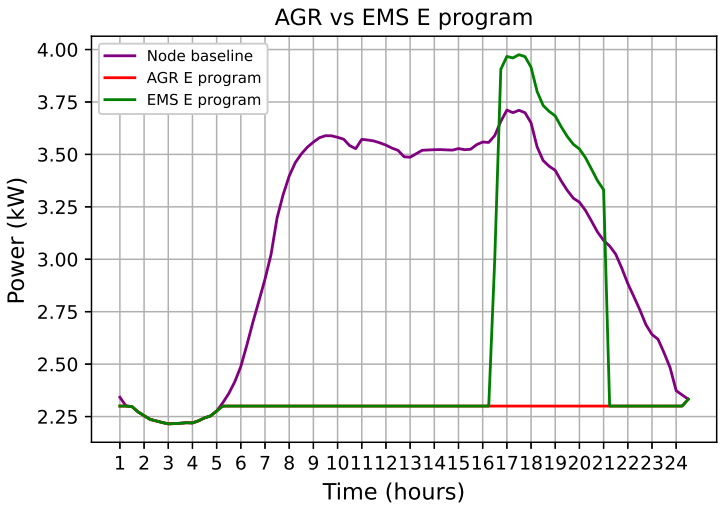
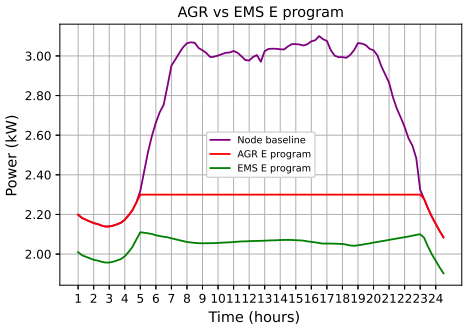


Figure 6.22: Energy program after EMS feasibility check in Winter

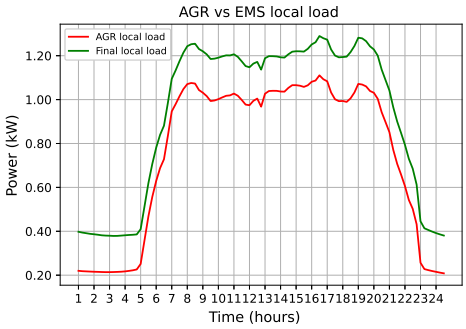
The next question that should be asked is what can be done to avoid this? The first step is for the AGR to detect that the bids are fully matched using the grid and remove them. The EMS does not use the grid exchange for charging any of the storage assets, so the AGR can just delete them and it will simply not fully match the flexibility request of the DSO. Then, the AGR can calculate the total revenue of the bids and do a rank filtering to see if there is another viable solution. The problem with these steps is that at the moment the AGR has no way to detect that some of the flexibility bids are being fully matched through the grid and thus would try to offer that flexibility bid to the DSO. It is also important that in this case, the EMS tries to fully match the flexibility bids while it could also by itself refuse to not fully match it. Finally, it is also important to note that the EMS matches the flexibility bids using a wide range of methods (storage, PV production, and grid). Each of these methods are based on a forecast which can have some deviations. For example, the PV production may be less than forecasted, which in this case would lead to a problem when serving the flexibility.

6.4.2. Simulation for the other seasons

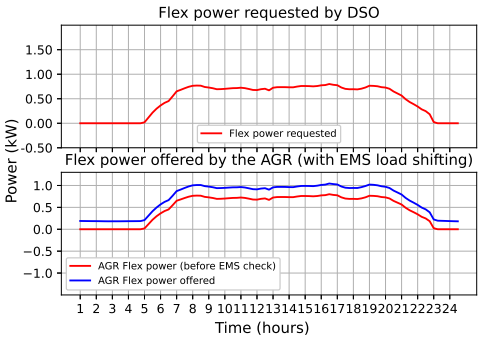
Figure 6.23 shows the local load, energy program, flexibility bids and change in operation after feasibility checks in the season of Fall. During this season the EMS does not use the grid to match flexibility as it can be seen in figure 6.23d, and thus does not create any further congestion. On the other hand, the EMS is over-serving flexibility. This can be seen in figure 6.23a where the EMS energy program shown in green is smaller than the AGR energy program or equivalently in figure 6.23b with the Final local load greater than the one proposed by the AGR. This does not make sense as the EMS has no gain in providing more energy. Furthermore, the DSO will not compensate for this additional flexibility as it did not request it. The reason for this excess flexibility cannot be justified but it is hypothesized that it might be an error in the EMS code. Take into consideration that the EMS code that this has been tested on is a first iteration and thus not a fully optimized version. Such deviation should be spotted by the AGR and corrected accordingly to avoid the EMS from unnecessarily using its assets. This is convenient for both the AGR, as it can use the power ”saved” for another day and the EMS, as it does not expose the users to possible discomforts.
Figure 6.24 and 6.25 show the local load, energy program, flexibility bids, and change in operation after feasibility checks in the season of Summer and Spring respectively. Likewise to the Fall season, the EMS over serves flexibility again which cannot be explained. Other than that, the EMS does not use the grid to match the flexibility and thus no complications arise from serving flexibility.



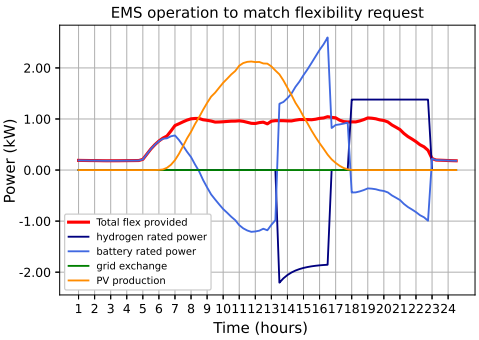
(a) Energy program after EMS feasibility check in Fall



(b) Local load after EMS's feasibility check in Fall

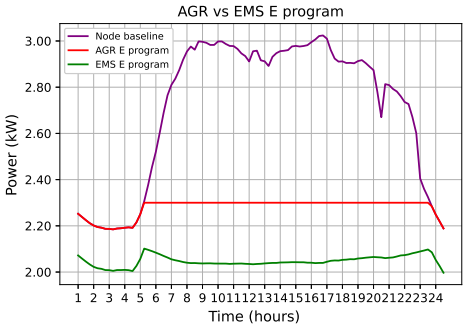


(c) Flexibility bids before and after the feasibility check in Fall

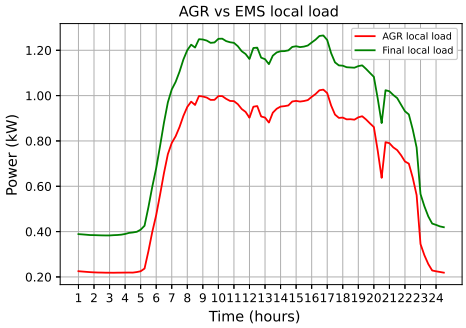


(d) EMS's way to match the flexibility request in Fall

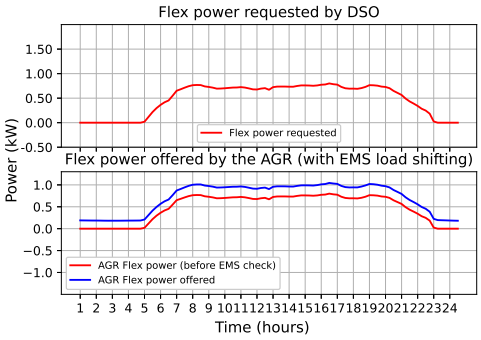
Figure 6.23: EMS's local load, energy program, flexibility bids, and change in operation for Fall



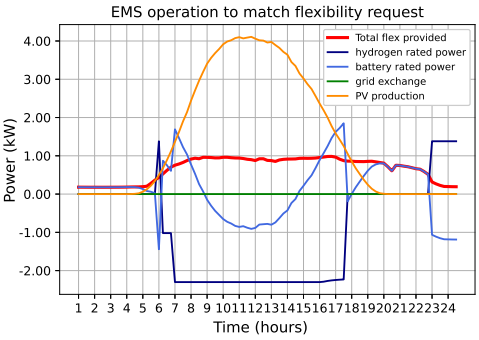
(a) Energy program after EMS feasibility check in Summer



(b) Local load after EMS's feasibility check in Summer

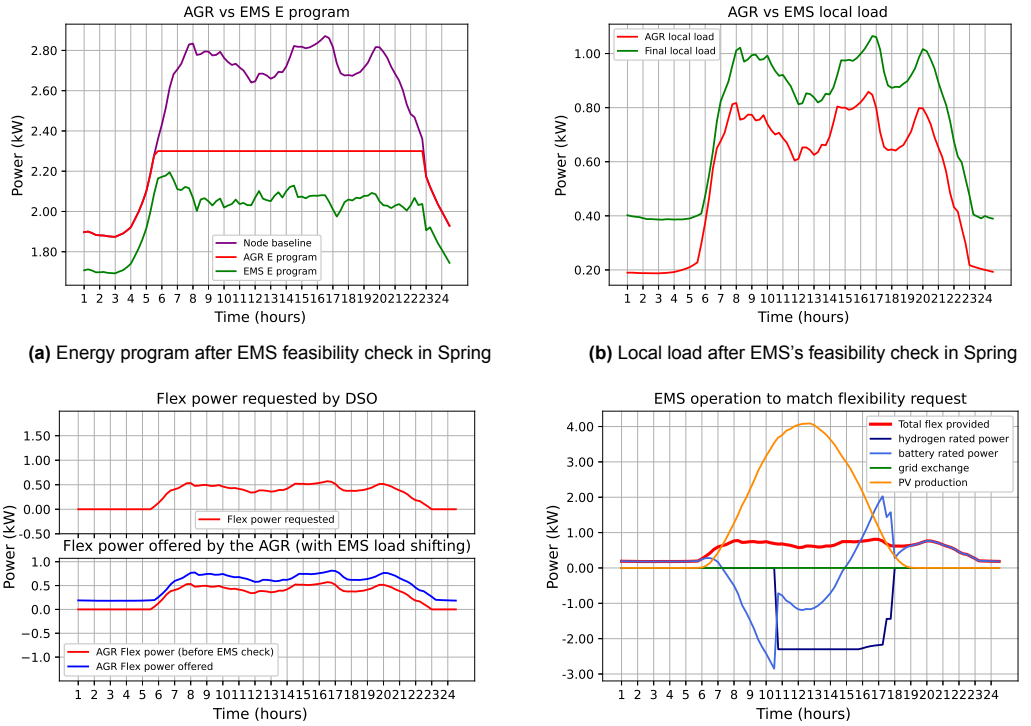


(c) Flexibility bids before and after the feasibility check in Summer



(d) EMS's way to match the flexibility request in Summer

Figure 6.24: EMS's local load, energy program, flexibility bids, and change in operation for Summer



(a) Energy program after EMS feasibility check in Spring

(b) Local load after EMS's feasibility check in Spring

(c) Flexibility bids before and after the feasibility check in Spring

(d) EMS's way to match the flexibility request in Spring

Figure 6.25: EMS's local load, energy program, flexibility bids, and change in operation for Spring

From these simulations, it is clear that the AGR still needs some modifications. In section 6.4.1 it was shown that the EMS was trying to fully match the flexibility by using the grid creating more congestion. The AGR program must be able to detect when the grid is being used inefficiently and correct the bids accordingly. In the Winter case, the AGR program wouldn't even need to double-check with the EMS, reducing the message exchange necessary between the two. The AGR program should also check when too much flexibility is being served. At the moment, the EMS tries to fully match the flexibility request which seems to create inconvenient situations shown for the season of Fall, Summer, and Spring. It is surprising how determined is the EMS to fully match the flexibility request. When developing the AGR program and more precisely the ranking system, it was envisioned that the EMS would rather reject flexibility bids than go out of its way to match it. From these simulations, it seems that it is not the case. Additionally, it looks like the EMS has some problems with delivering the correct amount of power. As seen for the seasons of Fall, Summer and Spring the EMS provides too much power. This could either be a problem with the EMS code or because of granularity. Either way, it is imperative that the AGR program is able to detect any excesses and correct the bids accordingly.

In this chapter the results of the simulations were presented. First, the EMS modes of operations were introduced and then the data available. The way the data is divided into seasons and how the baseline at the node and the flexibility request signal is created was then presented together with the Python's code structure.
The simulation results were then shown and discussed. One day of each season was run through the bidding process to give a clear understanding of how the process works in different situations. As expected, the hotter seasons requested less flexibility compared to the colder ones. As explained, only one day was chosen to be simulated because the AGR program tries to fully match the flexibility request. It is unlikely that the EMS will allow that and thus running the simulation over multiple consecutive days would have not yielded any insight into the program's operation. How the EMS' feasibility check is simulated was also explained. It was chosen that if the SoC of the storage asset reached 30% then then EMS would automatically refuse the bids. Additionally, certain time periods would be manually chosen to be refused by the EMS. These time periods were chosen during peak times as they are most likely the moments when the EMS wants to conserve energy.
The Ranking process was then introduced. As the EMS' feasibility check resulted in reduced revenue for the AGR, the Ranking process showed an alternative way for the AGR to reconsider its flexibility bids. By selecting an appropriate rank, it has been shown that there is the possibility to satisfy both the EMS and the AGR preferences. Keeping only high-valued bids, the AGR is able to get a greater revenue while providing less energy compared to the EMS' Final Energy program. Nonetheless, such a condition could only be satisfied if an appropriate rank was chosen. An automated script was created to find the optimal rank, using the ratios metric as conditions. Two main aspects were discussed. The first one is that the increment used in the rank automated script affected the findings. The smaller the increment, the more optimal the rank is. However, this is at the cost of lengthier computational time. The second is that ranking promotes the usage of flexibility during peak hours which is also the moment that the EMS has a more energy-conservative approach.
Finally, the AGR program's limitations were showcased. Starting with being unable to use the battery when charged and providing flexibility with load shifting. The short timeframe of operation of the AGR program was also discussed with its inability of planning ahead for future flexibility bids. Some simulation with a first iteration of the EMS code were carried out and showed some possible problems with integrating it in the flexibility trading.

7

Conclusion and recommendation

The current critical state of the electric grid was explained and the necessity of flexibility was argued. As discussed, standardization plays a major role in facilitating and accelerating the deployment of such technology, which was the goal of the protocol and code developed. While the protocol and code could be deployed for any type of system, a particular focus was directed at the residential level for two main reasons. The first one is that TGV is literally an aggregate of residential loads and the second is the lack of standardization in that domain thus the relevancy of the study.
The actors involved have complementary features that can be exploited to efficiently deploy flexibility but also conflicting interests that needed to be considered when designing the protocol and AGR program. This has led to the research question of how to create a AGR program that can trade flexibility in a responsible manner. This was tackled by implementing a protocol and AGR program that followed three structural requirements.
The first requirement is interoperability, to be compatible with as many systems as possible and increase the impact that one AGR can have. As explained in the report this was achieved by unburdening the EMS from as many tasks as possible and making the AGR do most of the computation. Nonetheless, some actions are still needed by the EMS, and while for an MPC control methodology, these actions are feasible, it is important in the future to study the compatibility of EMS using other control methodologies.
The second requirement is redundancy, to avoid possible miscommunications or errors. To do so the EMS decision was set as the top priority. This means that if the EMS would decide to change or decline all flexibility, then so be it. The reason for it is that the EMS is the direct owner of the assets (and has also direct control) and the AGR should not be able to decide without its agreement and shall follow its decisions. It would be interesting to study how likely it is for the EMS to go rogue and not follow the agreed energy program. Additionally, the protocol follows a similar strategy to the USEF Framework of sharing over and over the same information to continuously validate the message.
Finally, the last requirement is the plug-and-play. This was unlocked by creating a clear departmentalization of the code so that each section had a clear role and where they stand in the protocol. By doing so, anyone interested to modify or upgrade the code knows which part it needs to touch. As this is the first iteration of the protocol and code, it is interesting to see how future modifications will affect it and what limitations such structure has.
The validation through simulation has shown what a normal operation would look like, while some sensitivity analyses have displayed some constraints of the code. As explained, because of the timeline of this project with the one of TGV, the protocol and AGR program could not be thoroughly tested with a fully operational EMS (also in real-time) and it is thus imperative that more testing with the protocol counterparts is done to validate its efficiency.

References

[1] P. Palensky et al. *Oplossingsrichtingen voor congestie in middenspanningsnetten: de casus Buiksloterham-Zuid/Overhoeks Amsterdam*. Dutch. openresearch.amsterdam, 2022. DOI: 10.13140/RG.2.2.11476.68481.

[2] Benjamin Bayer et al. “The German experience with integrating photovoltaic systems into the low-voltage grids”. In: *Renewable Energy* 119 (Apr. 2018), pp. 129–141. ISSN: 0960-1481. DOI: 10.1016/j.renene.2017.11.045.

[3] R. Fonteijn et al. “Evaluating flexibility values for congestion management in distribution networks within Dutch pilots”. English. In: *The 7th International Conference on Renewable Power Generation*. The 7th International Conference on Renewable Power Generation (RPG2018), RPG 2018 ; Conference date: 26-09-2018 Through 27-09-2018. 2018. URL: https://www.wartsila.com/gbr/media/local-event/2018/08/26/default-calendar/the-7th-international-conference-on-renewable-power-generation.

[4] USEF Foundation. *USEF: The framework explained*. Tech. rep. May 2021.

[5] Giuseppe Prettico, Antonios Marinopoulos, and Silvia Vitiello. *Distribution System Operator Observatory 2020*. Feb. 2021. ISBN: 978-92-76-28430-7. DOI: 10.2760/311966.

[6] Werner van Westering and Hans Hellendoorn. “Low voltage power grid congestion reduction using a community battery: Design principles, control and experimental validation”. In: *Int. J. Electr. Power Energy Syst.* 114 (Jan. 2020), p. 105349. ISSN: 0142-0615. DOI: 10.1016/j.ijepes.2019.06.007.

[7] Aliander NV. *Flexibility from residential power consumption: a new market filled with opportunities*. Tech. rep. 2016.

[8] Stamatios Chondrogiannis et al. *Local electricity flexibility markets in Europe*. Oct. 2022. ISBN: 978-92-76-56156-9. DOI: 10.2760/9977.

[9] TKI Urban Energy and Topsector Energie. *In-Home Energy Flexibility Protocols*. Tech. rep. June 2020.

[10] OpenADR Alliance. *OpenADR 2.0 Profile Specification B profile*. Tech. rep. OpenADR Alliance, November 2015.

[11] USEF Foundation and OpenADR. *OpenADR Alliance, USEF Foundation Partnership Takes first step to standardize flexibility programs*. October 2016. URL: https://www.usef.energy/app/uploads/2017/01/OADR_USEF_Oct16.pdf.

[12] Digvijay Gusain, Milos Cvetkovic, and Peter Palensky. “Quantification of Operational Flexibility from a Portfolio of Flexible Energy Resources”. In: *International Journal of Electrical Power and Energy Systems* 141 (2021), p. 107466. ISSN: 0142-0615. DOI: 10.1016/j.ijepes.2021.107466.

[13] Lukas Bank et al. *Integrating Energy Flexibility in Production Planning and Control - An Energy Flexibility Data Model-Based Approach*. Hannover, Germany: Hannover : publish-Ing., 2021. URL: https://www.repo.uni-hannover.de/handle/123456789/11336.

[14] *24/7 Energy Lab: local, autonomous and CO2 free*. [Online; accessed 26. Apr. 2023]. Apr. 2023. URL: https://www.tudelft.nl/en/delft-integraal/articles/dec-2021-nieuwe-energie/24-7-energy-lab-local-autonomous-and-co2-free.

[15] Evgenia Mechleri, Bogdan Dorneanu, and Harvey Arellano-Garcia. “A Model Predictive Control-Based Decision-Making Strategy for Residential Microgrids”. In: *Eng* 3.1 (Feb. 2022), pp. 100–115. ISSN: 2673-4117. DOI: 10.3390/eng3010009.

[16] luti. mambweni. “What is an EMS - Energy Management System? | METRON”. In: *Metron* (Jan. 2023). URL: https://www.metron.energy/blog/ems-energy-management-system-definition.

[17] *N-SIDE | Forecasting and optimization solutions for power grids and markets*. [Online; accessed 28. Apr. 2023]. Apr. 2023. URL: https://energy.n-side.com.

[18] Nikolaos G. Paterakis, Ozan Erdin, and Joo P. S. Catalo. “An overview of Demand Response: Key-elements and international experience”. In: *Renewable Sustainable Energy Rev.* 69 (Mar. 2017), pp. 871–891. ISSN: 1364-0321. DOI: 10.1016/j.rser.2016.11.167.

[19] *EUR-Lex - 32019L0944 - EN - EUR-Lex*. [Online; accessed 1. May 2023]. May 2023. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex%5C%3A32019L0944.

[20] Selina Kerscher and Pablo Arboleya. “The key role of aggregators in the energy transition under the latest European regulatory framework”. In: *Int. J. Electr. Power Energy Syst.* 134 (Jan. 2022), p. 107361. ISSN: 0142-0615. DOI: 10.1016/j.ijepes.2021.107361.

[21] Xiaoxing Lu et al. “Fundamentals and business model for resource aggregator of demand response in electricity markets”. In: *Energy* 204 (Aug. 2020), p. 117885. ISSN: 0360-5442. DOI: 10.1016/j.energy.2020.117885.

[22] *How does GOPACS work? - GOPACS*. [Online; accessed 26. Apr. 2023]. May 2022. URL: https://en.gopacs.eu/about-gopacs-copy.

[23] USEF Foundation. *USEF Flexibility Trading Protocol Specifications*. Tech. rep. January 2020.

[24] USEF Foundation. *USEF: The framework explained*. Tech. rep. November 2015.

[25] Hayden M. Reeve et al. *Distribution System Operator with Transactive (DSO+T) Study: Volume 1 (Main Report)*. [Online; accessed 23. May 2023]. Jan. 2022. DOI: 10.2172/1842485.

[26] Dasom Lee, David J. Hess, and Himanshu Neema. “The challenges of implementing transactive energy: A comparative analysis of experimental projects”. In: *Electricity Journal* 33.10 (Dec. 2020), p. 106865. ISSN: 1040-6190. DOI: 10.1016/j.tej.2020.106865.

[27] *Transactive Energy | PNNL*. [Online; accessed 24. April 2023]. 2023. URL: https://www.pnnl.gov/explainer-articles/transactive-energy.

[28] Tobias Goebel. “What is MQTT?” In: *Twilio Blog* (Mar. 2023). URL: https://www.twilio.com/blog/what-is-mqtt.

[29] *OpenRemote | The 100% Open Source IoT Platform*. [Online; accessed 19. Jun. 2023]. Mar. 2023. URL: https://www.openremote.io.

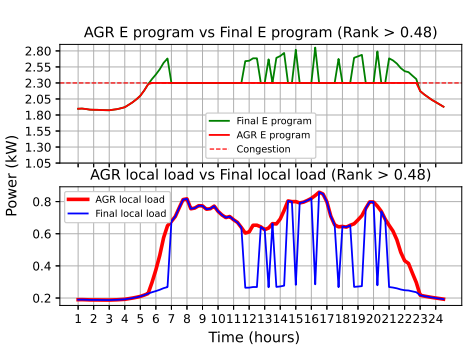
[30] *Data Platform – Open Power System Data*. [Online; accessed 3. May 2023]. May 2023. URL: https://data.open-power-system-data.org/time_series.

A

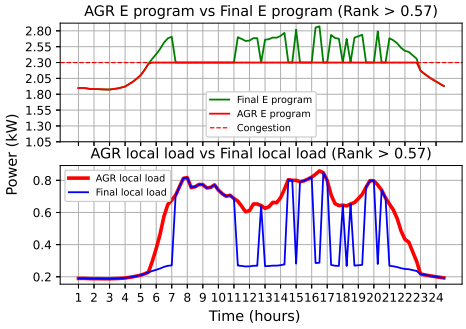
Additional figures

Additional figures related section 6.2.5.

AGR energy program with maximal price deviation:



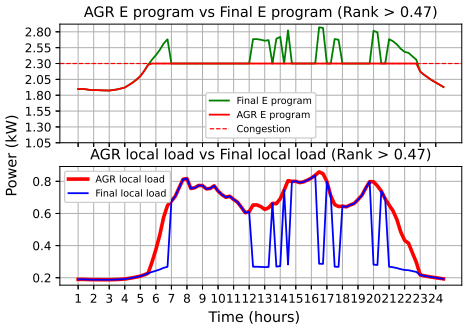
(a) Optimal AGR energy program with maximal price deviation



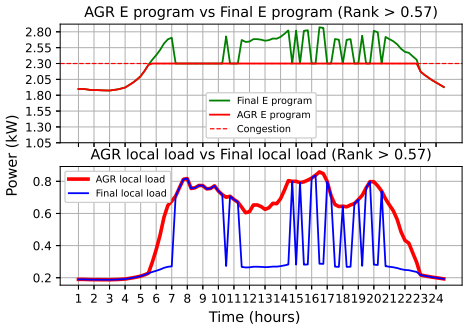
(b) Suboptimal AGR energy program with maximal price deviation

Figure A.1: AGR energy program with maximal price deviation

AGR energy program with mean price deviation:



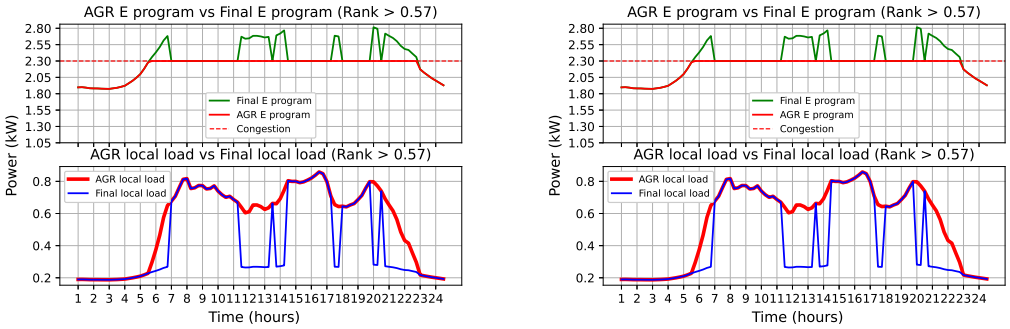
(a) Optimal AGR energy program with mean price deviation



(b) Suboptimal AGR energy program with mean price deviation

Figure A.2: AGR energy program with mean price deviation

AGR energy program with minimal price deviation:



(a) Optimal AGR energy program with minimal price deviation **(b)** Suboptimal AGR energy program with minimal price deviation

Figure A.3: AGR energy program with minimal price deviation

B

AGR program code

This section contains the codes of the AGR program, Automated Ranking, and plots.

Here below is main script of the AGR program:

```
# -*- coding: utf-8 -*-
"""
Created on Wed Jan 25 13:28:45 2023

@author: carlo
"""
from pyomo.environ import *
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
import math

# import functions:
import Flexibility_FUNC as FR
import Flexibility_FUNC_V2 as FR2
import Flexibility_FUNC_V3 as FR3


# SELECTION ____________________________________________________________________________

df = pd.read_csv('results.csv')
df_price = pd.read_csv('price.csv')

## Divide the data into different seasons
Winter1 = df.iloc[32162:35137,0:12].reset_index(drop=True)
Winter2 = df.iloc[0:5762,0:12].reset_index(drop=True)
Winter = Winter1.append(Winter2, ignore_index=True) # c=2494

Spring = df.iloc[5762:14593,0:12].reset_index(drop=True) # c=4702

Summer = df.iloc[14594:23425,0:12].reset_index(drop=True) # c=3262

Fall = df.iloc[23426:32162,0:12].reset_index(drop=True) # c=3166

# constant used to access values for each seasons
c=4702
EMS = Spring.iloc[c:c+95,1:12].reset_index(drop=True)

price = df_price.iloc[c:c+95,4].reset_index(drop=True)
penalty = price + 50

congestion_limit= 0.230*10
grid_exchange = EMS.iloc[:,0]*10 #This grid exchange is created to fake exchange between the
    TGV and the grid.
```

```
EMS['external'] = grid_exchange



# Computation of the Flexibility request
FR_PTU = FR.DA_FR(EMS.external,congestion_limit).fillna(0)

# Optimization results
SoC_change, PshFlex, AGR_E_prog = FR.Flex_calc(EMS, FR_PTU)


# Calculates the difference between the requested flex by the DSO and the flex offered by the AGR
PshFlex['TotFlexUnservedAGR'], PshFlex['TotFlexServedAGR'] = FR.Difference_calc(PshFlex.TotFlex, FR_PTU.Flex_Request, PshFlex.TotFlex)


# Calculates the revenue from the AGR proposed bids
Revenue_AGR, Revenue_AGR_sum = FR.Financial_calc(PshFlex.TotFlex, price)

# Calculates the ranks of the revenue of the AGR proposed bids
rank = FR.Ranking(Revenue_AGR)

# DataFrames are created for each case (manually rejected bids)
Final_Psh = pd.DataFrame(index = AGR_E_prog.index, columns=['program','local'])
Final_Psh.program = AGR_E_prog.program
Final_Psh.local = AGR_E_prog.local

# DataFrames are created for each case (SoC rejected bids)
Final_PshManual = pd.DataFrame(index = AGR_E_prog.index, columns=['program','local'])
Final_PshManual.program = AGR_E_prog.program
Final_PshManual.local = AGR_E_prog.local


# DataFrames are created for each case (SoC rejected bids)
Final_PshAuto = pd.DataFrame(index = AGR_E_prog.index, columns=['program','local'])
Final_PshAuto.program = AGR_E_prog.program
Final_PshAuto.local = AGR_E_prog.local

#-------------------------------------

## SIMULATING THE EMS REJECTION


## "Automatice rejection"
SoC_limit_batt = 0.3
SoC_limit_H2 = 0.3
# Simulates the automatic rejection of flexibility bids by the EMS when the SoC of the storage assets is below a certain limit.
for t in range(0,max(EMS.index)):
    if SoC_change.battery_soc_new.iloc[t] <= SoC_limit_batt:
        Final_PshAuto.program.iloc[t] = EMS.external.iloc[t] #+ PshFlex.batt_power[t]
        Final_PshAuto.local.iloc[t] = EMS.load_forecast.iloc[t] #- PshFlex.batt_power[t]
    if SoC_change.H2_soc_new.iloc[t] <= SoC_limit_H2:
        Final_PshAuto.program.iloc[t] = EMS.external.iloc[t] #+ PshFlex.H2_power[t]
        Final_PshAuto.local.iloc[t] = EMS.load_forecast.iloc[t] #- PshFlex.H2_power[t]

# for t in range(0,max(EMS.index)):
#     if SoC_change.battery_soc_new.iloc[t] <= SoC_limit_batt:
#         Final_Psh.program.iloc[t] = EMS.external.iloc[t] #+ PshFlex.batt_power[t]
#         Final_Psh.local.iloc[t] = EMS.load_forecast.iloc[t] #- PshFlex.batt_power[t]
#     if SoC_change.H2_soc_new.iloc[t] <= SoC_limit_H2:
#         Final_Psh.program.iloc[t] = EMS.external.iloc[t] #+ PshFlex.H2_power[t]
#         Final_Psh.local.iloc[t] = EMS.load_forecast.iloc[t] #- PshFlex.H2_power[t]


# # Manual rejection
# for i in range(56,95): ## The EMS rejects the flex bids for those time moments
#     Final_PshManual.program.iloc[i] = EMS.external.loc[i]
#     Final_PshManual.local.iloc[i] = EMS.load_forecast.loc[i]

```

```
for i in range(17,44):
    Final_PshManual.program.iloc[i] = EMS.external.loc[i]
    Final_PshManual.local.iloc[i] = EMS.load_forecast.loc[i]

# # Manual rejection
# for i in range(56,95): ## The EMS rejects the flex bids for those time moments
#     Final_Psh.program.iloc[i] = EMS.external.loc[i]
#     Final_Psh.local.iloc[i] = EMS.load_forecast.loc[i]

for i in range(17,40):
    Final_Psh.program.iloc[i] = EMS.external.loc[i]
    Final_Psh.local.iloc[i] = EMS.load_forecast.loc[i]

PshFlex['TotFlexUnservedUnrank'], PshFlex['TotFlexServedUnrank'] = FR.Difference_calc(
    Final_Psh.program, AGR_E_prog.program, PshFlex.TotFlex)

# Ranking filtering process
    ----------------------------------------------------------------------------------

execfile("Ranking_Automated.py")

## EMS deviation
    simulation_____________________________________________________________________________
## This section simulates the deviations in the Final Energy program. The deviations are
    manually inserted.

# FInal_Psh3 is the EMS's final E prog with deviations
Final_Psh3 = pd.DataFrame(index = AGR_E_prog.index, columns=['program','local'])
Final_Psh3.program = Final_Psh.program
Final_Psh3.local = Final_Psh.local

# Changes in power flex provided if the EMS deviations use the grid
PshFlex3 = pd.DataFrame(index = AGR_E_prog.index, columns=['TotFlex'])
PshFlex3.TotFlex = PshFlex.TotFlex

for i in range(44,48):
    Final_Psh3.program.iloc[i] = Final_Psh3.program.iloc[i] + 0.008
    Final_Psh3.local.iloc[i] = Final_Psh3.local.iloc[i] #+ 0.008
    PshFlex3.TotFlex.iloc[i] = PshFlex3.TotFlex.iloc[i] - 0.008


for i in range(42,44):
    Final_Psh3.program.iloc[i] = Final_Psh3.program.iloc[i] + 0.008
    Final_Psh3.local.iloc[i] = Final_Psh3.local.iloc[i] #+ 0.008
    PshFlex3.TotFlex.iloc[i] = PshFlex3.TotFlex.iloc[i] - 0.008

for i in range(67,74):
    Final_Psh3.program.iloc[i] = Final_Psh3.program.iloc[i] + 0.008
    Final_Psh3.local.iloc[i] = Final_Psh3.local.iloc[i] #+ 0.008
    PshFlex3.TotFlex.iloc[i] = PshFlex3.TotFlex.iloc[i] - 0.008



## Categorizing
    results_____________________________________________________________________________________________


FlexAnalysis = pd.DataFrame([[0,0,0]], columns=['CumFlex','CumFlexBatt','CumFlexH2'])

FlexAnalysis['CumFlex'] = np.cumsum(PshFlex['batt_power']).max() + np.cumsum(PshFlex['
    H2_power']).max()
FlexAnalysis['CumFlexBatt'] = np.cumsum(PshFlex['batt_power']).max()
FlexAnalysis['CumFlexH2'] = np.cumsum(PshFlex['H2_power']).max()
FlexAnalysis['CumUnservedFlex'] = np.cumsum(PshFlex['TotFlexUnservedRank']).max()
FlexAnalysis['CumServedFlex'] = np.cumsum(PshFlex['TotFlexServedRank']).max()

# plots
execfile("plot_v2.py")

print('\n \n!_____________CODE FINISHED_____________!')
```

Here below are the functions used by the AGR program:

```
# -*- coding: utf-8 -*-
"""
Created on Mon Feb 13 14:33:29 2023

@author: carlo
"""
import pandas as pd
import numpy as np
from pyomo.environ import *
import matplotlib.pyplot as plt

## Calculates bids based on an optimization. The flexibility request (FR) is minimized by bidding power (Psh). obj. function = min{FR-Psh}.
def Flex_calc(EMS,FR_PTU): #Inputs: EMS's baseline and Flexibility request.
    ## CREATING MODEL, SET AND PARAM
    model = AbstractModel()

    # Creating sets for each decision variable
    model.BATTERY = Set()
    model.HYDROGEN = Set()

    # Creating the different variables that will be used for the constraints
    model.BatteryMinPower = Param(model.BATTERY)
    model.BatteryMaxPower = Param(model.BATTERY)
    model.BatteryCapacity = Param(model.BATTERY)
    model.BatterySoC = Param(model.BATTERY)
    model.BatteryPrated = Param(model.BATTERY)
    model.HydrogenMinPower = Param(model.HYDROGEN)
    model.HydrogenMaxPower = Param(model.HYDROGEN)
    model.HydrogenCapacity = Param(model.HYDROGEN)
    model.HydrogenSoC = Param(model.HYDROGEN)
    model.HydrogenPrated = Param(model.HYDROGEN)

    # Creating dataframes that will be used for the optimization:
    PshFlex = pd.DataFrame(index=EMS.index) # Amount of power provided at each PTU
    SoC = pd.DataFrame(index=EMS.index) # The SoC of the different assets
    SoC_change = pd.DataFrame(index=EMS.index) # The change in SoC of each assets
    P_rated_EMS = pd.DataFrame(index=EMS.index) # How much the asset is already in use during the request of power
    AGR_E_prog = pd.DataFrame(index=EMS.index) # The energy program proposed by the AGR.


    # Assigning the values of the EMS to the respective variables.
    # Note: Soc used for defining the parameters of the optimization and SoC_change for saving the optimization results.

    SoC['battery_soc'] = EMS['battery_soc_actual']
    SoC['H2_soc'] = EMS['hydrogen_soc_actual']
    SoC_change['battery_soc_new'] = SoC['battery_soc'] #This colums will be the altered SoC after the bidding has been done.
    SoC_change['H2_soc_new'] = SoC['H2_soc'] #This colums will be the altered SoC after the bidding has been done.
    P_rated_EMS['Battery_Prated'] = -EMS['battery_actual']
    P_rated_EMS['H2_disPrated'] = -EMS['hydrogen_actual']

    # These are just starting value to initaliaze the model but will be changed later.
    PshFlex['batt_power'] = 0 # power used for flex bid
    PshFlex['H2_power'] = 0 # power used for flex bid
    FR = 0 # request of power.


    # # Setting up the paramaters. A dictionary type of strucuture is used
    # # so that the values can be continuously updated:

    data = {None: { #Somehow the Set will not properly create if I don't insert this None. I do not know the reason why :(
        'BATTERY': {None: [1]},
        'HYDROGEN': {None: [1]},
```

```
        'BatteryMaxPower': {1: 4},
        'BatteryMinPower': {1: 0},
        'BatteryCapacity': {1: 15.4},
        'BatterySoC': {1: SoC.iloc[0,0]},
        'BatteryPrated': {1: P_rated_EMS.iloc[0,0]},
        'HydrogenMinPower': {1: 0},
        'HydrogenMaxPower': {1: 2.3},
        'HydrogenCapacity': {1: 1998},
        'HydrogenSoC': {1: SoC.iloc[0,1]},
        'HydrogenPrated': {1: P_rated_EMS.iloc[0,1]}
    }  }


    # # Decision variable
    model.BatteryPsh = Var(model.BATTERY, within=Reals, initialize=0)
    model.HydrogenPsh = Var(model.HYDROGEN, within=Reals, initialize=0)


    # Objective Function
    def objective_rule(model): # For a flexibility request (FR) the power (Psh) of the assets is used to match it.
        return FR - (sum(model.BatteryPsh[i] for i in model.BATTERY) + sum(model.HydrogenPsh[j] for j in model.HYDROGEN))

    # Constraints

    def MaxPowerRated_rule_BATT(model,i): # Max rated power limit
        return  model.BatteryPsh[i]  <= model.BatteryMaxPower[i]

    def MinPowerRated_rule_BATT(model,i): # Min rated power limit
        return  model.BatteryMinPower[i]  <= model.BatteryPsh[i]

    def MaxCapacityLimits_rule_BATT(model,i): # Checks that the power flex is within the limits of the storage (discharge limit).
        return model.BatteryPsh[i] <= model.BatteryCapacity[i]*model.BatterySoC[i]*4

    def MaxPowerAvailable_rule_BATT(model,i): # Checks how much power the asset is already using at that moment.
        return model.BatteryPsh[i] <= model.BatteryMaxPower[i] - model.BatteryPrated[i]

    def MinPowerRated_rule_HYDRO(model,i): # Min rated power limit
        return  model.HydrogenMinPower[i] <= model.HydrogenPsh[i]

    def MaxPowerRated_rule_HYDRO(model, i): # Max rated power limit
        return  model.HydrogenPsh[i]  <= model.HydrogenMaxPower[i]

    def MaxCapacityLimits_rule_HYDRO(model, i): # Checks that the power flex is within the limits of the storage (discharge limit).
        return model.HydrogenPsh[i] <= model.HydrogenCapacity[i]*model.HydrogenSoC[i]*4

    def MaxPowerAvailable_rule_HYDRO(model, i):
        if (model.HydrogenPrated[i] <= 0): #valid only when the H2 asset is is used. Chargin and discharging are decoupled and can happen at the same time. # Checks how much power the asset is already using at that moment.
            return model.HydrogenPsh[i] <= model.HydrogenMaxPower[i] - model.HydrogenPrated[i] #valid only when the H2 asset is is used. Chargin and discharging are decoupled and can happen at the same time. # Checks how much power the asset is already using at that moment.
        else:
            return model.HydrogenPsh[i] <= model.HydrogenMaxPower[i]

    def LimitPshFlex_rule_HYDRO(model,i): # Makes sure that the optimization does not provide more power than what the FR needs. (minimization wil try to reduce it as much as possible)
        return sum(model.HydrogenPsh[i] for i in model.HYDROGEN) <= FR

    def LimitPshFlex_rule_BATT(model,i): # Makes sure that the optimization does not provide more power than what the FR needs. (minimization wil try to reduce it as much as possible)
        return sum(model.BatteryPsh[i] for i in model.BATTERY) <= FR

```

```
120
121     # Activating the constraints
122     model.MaxPowerRated_BATT = Constraint(model.BATTERY, rule=MaxPowerRated_rule_BATT)
123     model.MinPowerRated_BATT = Constraint(model.BATTERY, rule=MinPowerRated_rule_BATT)
124     model.MaxCapacityLimits_BATT = Constraint(model.BATTERY, rule=MaxCapacityLimits_rule_BATT
            )
125     model.MaxPowerAvailable_BATT = Constraint(model.BATTERY, rule=MaxPowerAvailable_rule_BATT
            )
126     model.MaxPowerRated_HYDRO = Constraint(model.HYDROGEN, rule=MaxPowerRated_rule_HYDRO)
127     model.MinPowerRated_HYDRO = Constraint(model.HYDROGEN, rule=MinPowerRated_rule_HYDRO)
128     model.MaxCapacityLimits_HYDRO = Constraint(model.HYDROGEN, rule=
            MaxCapacityLimits_rule_HYDRO)
129     model.MaxPowerAvailable_HYDRO = Constraint(model.HYDROGEN, rule=
            MaxPowerAvailable_rule_HYDRO)
130     model.LimitPshFlex_BATT = Constraint(model.BATTERY, rule=LimitPshFlex_rule_BATT)
131     model.LimitPshFlex_HYDRO = Constraint(model.HYDROGEN, rule=LimitPshFlex_rule_HYDRO)
132     # model.LimitPshFlex = Constraint(model.HYDROGEN, model.BATTERY, rule=LimitPshFlex_rule)
133
134     model.PowerProvided = Objective(rule=objective_rule, sense=minimize)
135     instance = model.create_instance(data)
136
137
138     # For loop for each PTU is created. Within the for loop the model is run through the
            optimization and
139     # the assets change of state is updated for the following PTU. (i.e. battery discharged =
             SoC decreases)
140
141     for t in range(0,len(EMS.index)-1):
142
143         # The values of the SoC and rated power are updated as the for loop iterates.
144         data = {None: { #Somehow the Set() will not create propery if I don't insert this
                None. I do not know the reason why :(
145             'BATTERY': {None: [1]},
146             'HYDROGEN': {None: [1]},
147             'BatteryMaxPower': {1: 4},
148             'BatteryMinPower': {1: 0},
149             'BatteryCapacity': {1: 15.4},
150             'BatterySoC': {1: SoC.iloc[t,0]},
151             'BatteryPrated': {1: P_rated_EMS.iloc[t,0]},
152             'HydrogenMinPower': {1: 0},
153             'HydrogenMaxPower': {1: 2.3},
154             'HydrogenCapacity': {1: 1998},
155             'HydrogenSoC': {1: SoC.iloc[t,1]},
156             'HydrogenPrated': {1: P_rated_EMS.iloc[t,1]}
157         }  }
158
159         # The flexibility requests changes at every iteration.
160         FR = FR_PTU.loc[t,'Flex_Request']
161
162         # Obejective goal is set
163         model.PowerProvided = Objective(rule=objective_rule, sense=minimize)
164
165         # The model is created with the updated data
166         instance = model.create_instance(data)
167
168         # Checks that the battery asset is available for usage (not charging). The H2 can
                work also during charging because it is formed of a compressor and fuel cell.
169         if (P_rated_EMS.iloc[t,0] < 1e-4): # If assets is being charged or low in charge then
                 disable decision variable.
170             instance.BatteryPsh[1].fix(0)
171         elif (P_rated_EMS.iloc[t,0] > 1e-4):
172             instance.BatteryPsh[1].unfix()
173
174         # instance.HydrogenPsh[1].fix(0)
175
176
177         # The optimization is runned and solved.
178         opt = SolverFactory('glpk')
179         opt.solve(instance)
180         print('\n \n working... \n \n ')
181
```

```
182
183         for i in range(0,2):
184             if (instance.HydrogenPsh[1].value == FR):
185                 if i == 0:
186                     PshFlex.iloc[t,i] = 0
187                 if i == 1:
188                     PshFlex.iloc[t,i] = instance.HydrogenPsh[1].value
189             else:
190                 if i == 0:
191                     if((instance.HydrogenPsh[1].value + instance.BatteryPsh[1].value) <= FR):
192                         PshFlex.iloc[t,i] = instance.BatteryPsh[1].value
193                     elif((instance.HydrogenPsh[1].value + instance.BatteryPsh[1].value) > FR)
                            :
194                         PshFlex.iloc[t,i] = FR - instance.HydrogenPsh[1].value
195                 if i == 1:
196                     PshFlex.iloc[t,i] = instance.HydrogenPsh[1].value
197
198
199             #Computes the change in SoC happening due to the discharge
200             SoC_change.loc[t,i] = ( (PshFlex.iloc[t,i]) * 0.25 )/instance.HydrogenCapacity[1]
                  #Computes the change in SoC happening due to the discharge
201
202
203             if SoC_change.iloc[t,i+2] >= 0: # This if loop is to calculate the SoC change is
                    load shedding is the chosen option
204                 if t == 0: #At first it uses the value of the original SoC
205                     SoC_change.iloc[t,i] = SoC.iloc[t,i] - SoC_change.iloc[t,i+2] + (SoC.iloc
                            [t+1,i] - SoC.iloc[t,i]) #The SoC is updated to the new value.
206                     SoC_change.iloc[t+1,i] = SoC_change.iloc[t,i] #The SoC of the next value
                            is updated for the next iteration/optimization
207                 else: #And then it starts creating its own SoC
208                     SoC_change.iloc[t,i] = SoC_change.iloc[t-1,i] - SoC_change.iloc[t,i+2] +
                            (SoC.iloc[t+1,i] - SoC.iloc[t,i]) #The SoC is updated to the new
                            value.
209                     SoC_change.iloc[t+1,i] = SoC_change.iloc[t,i]
210
211
212     # Creates traces of the assets SoC with the deviations from their original planned
            operation (for plotting)
213     SoC_change['battery_shift'] = SoC['battery_soc'] - SoC_change.iloc[:,2]
214     SoC_change['H2_shift'] = SoC['H2_soc'] - SoC_change.iloc[:,3]
215
216     # Adds the power of the provided flex at each PTU to have the total flexibility power.
217     PshFlex['TotFlex'] = PshFlex['batt_power'] + PshFlex['H2_power']
218
219     # Creates trace of the AGR energy program
220     AGR_E_prog['program'] = EMS.external - PshFlex.TotFlex
221     AGR_E_prog['local'] = EMS.load_forecast + PshFlex.TotFlex
222
223     return SoC_change, PshFlex, AGR_E_prog
224
225
226
227 def PV_curtailment(batt_p, H2_p, H2_on, load, pv_prod):# Inputs: The battery and H2 rated
        power, the local load and PV production
228     ## Calculates the amount of PV production that is curtailed (or just unused)
229     temp_batt = pd.DataFrame(index=load.index)
230     for i in range(0,len(load.index)):  ## Saves in dataframe the moment when the battery is
            charging
231         if batt_p.loc[i] < 0:
232             temp_batt.loc[i,'batt'] = -batt_p.loc[i]
233     temp_batt.batt = temp_batt.batt.fillna(0)
234
235     temp_H2 = pd.DataFrame(index=load.index)
236     for i in range(0,len(load.index)): ## Saves in dataframe the moment when the H2 is
            charging
237         if H2_on.loc[i] == 1 :
238             temp_H2.loc[i,'H2'] = H2_p.loc[i]
239     temp_H2.H2 = temp_H2.H2.fillna(0)
240
241     # Checks the difference between the PV production and the total load (batt+H2+local). If
```

```
        any remains then there is PV overproduction.
    pv_curtail = pv_prod - (temp_H2.H2 + temp_batt.batt + load)
    return pv_curtail


def Financial_calc(Power, price): # Inputs: Power served/unserved and price.
    ## Calculates the financial turnover. This can be used for calculating the revenue (power sold times the price)
    # but also financial penalties (power not supplied times the penalty) and financial turnovers for other purposes.
    Financial_turnover = pd.DataFrame(index=Power.index, columns=['turnover'])
    Financial_turnover = Power*price # Multiplies the provided flex with the price
    Financial_turnover_sum = Financial_turnover.cumsum().max() # Cumulative sum of the financial turnover
    return Financial_turnover, Financial_turnover_sum


def Difference_calc(AGR_E_prog, EMS_E_prog, Psh): #Inputs: E prog #1, E prog #2, Power Flex Offered
    ## Calculates the difference: E prog #2 - E prog #1. Return the amount of power that matches but also does not match.
    E_prog = pd.DataFrame(index=AGR_E_prog.index, columns=['Unserved','Served'])
    for i in range(0,len(AGR_E_prog.index)):
        if(EMS_E_prog.iloc[i] >= AGR_E_prog.iloc[i]): # Checks if there is a difference between the two E program.
            E_prog.Unserved.loc[i] = (EMS_E_prog.iloc[i] - AGR_E_prog.iloc[i]) # The amount of power planned but not served
    E_prog.Served = Psh - E_prog.Unserved # The amount of power reduced
    E_prog = E_prog.fillna(0)
    return E_prog.Unserved, E_prog.Served


def Ranking(Revenue,Power=None): #Inputs: Revenue, Dateframe to append the rank to. If none is assigned then it will return as an independent dataframe.
    ## Gives a ranking based on the revenue (power time price). The rank is calculated as: revenue at PTU divided by the greatest revenue of the whole set.
    # This should help prioritize certain bids over others
    Rank = pd.DataFrame(index=Revenue.index, columns=['rank'])
    Rank['rank'] = Revenue/Revenue.max() # Ranks revenue based of the max achievable.
    if Power is None:
        return Rank
    elif Power is not None: #Gives the option to have the rank appended directly to the flex power dataframe
        Rank = pd.merge(Power,Rank,left_index=True,right_index=True)
        return Rank

def Rank_analysis(PshFlex, price, penalty): #Inputs: The power unserved and served with no rank and with rank and the price and penalty.
    ## Calculates the difference in revenue, loss and volume served between the ranked and unranked energy programs
    Rank_analysis_temp = pd.DataFrame()
    lossUnRanked, lossUnRanked_sum = Financial_calc(PshFlex.TotFlexUnservedUnrank,penalty)
    revUnRanked, revUnRanked_sum = Financial_calc(PshFlex.TotFlexServedUnrank,price)

    lossRanked, lossRanked_sum = Financial_calc(PshFlex.TotFlexUnservedRank,penalty)
    revRanked, revRanked_sum = Financial_calc(PshFlex.TotFlexServedRank,price)

    # Somehow if the _sum variable is used it creates problem when appending it to a Dataframe. To fix this the cumulative sum of the revenue and loss is computed again.
    Rank_analysis_temp['lossUnRanked'] = lossUnRanked.cumsum()
    Rank_analysis_temp['revUnRanked'] = revUnRanked.cumsum()
    Rank_analysis_temp['lossRanked'] = lossRanked.cumsum()
    Rank_analysis_temp['revRanked'] = revRanked.cumsum()

    # Only the max is taken as we are interested in the total revenue and loss
    Rank_analysis = Rank_analysis_temp.max()

    return Rank_analysis


def EMS_package(AGR_E_prog,PshFlex): #Inputs: the EMS dataframe, the AGR dataframe (changes
```

```
    after flex bids), the Psh (amount of power bidded for flexibilty)
    ## Creates a dataframe with the information that need to be passed on to the EMS. Namely:
         the change in local load and grid exchange
    AGR_proposition = pd.DataFrame(index=PshFlex.index)

    AGR_proposition['local'] = AGR_E_prog.local
    AGR_proposition['external'] = AGR_E_prog.program
    AGR_proposition['trade'] = PshFlex.TotFlex

    return AGR_proposition



## NOT NEEDED ONCE THE SYSTEM IS UP AND RUNNING____________________________________________________
# These functions are created just to simulate inputs and outputs so that the AGR code can be tested
#I.e. Flex request is sent by DSO, or deviations in the actual power that is eventually provided by the EMS are due to ICT malfunctions or bad forcast.


## Simulates the DSO forecast of when the congestion happens and creates a flex request
def DA_FR(forecast,congestion_limit):
    FR_PTU = pd.DataFrame(index=forecast.index, columns=['P_Available','Flex_Request'])
    for i in range(0,len(forecast)):
        if(forecast.iloc[i] >= congestion_limit): # grid use is above grid limit so the flex is required
            FR_PTU.loc[i,'Flex_Request'] = forecast.iloc[i] - congestion_limit
        elif(forecast.iloc[i] < congestion_limit): # grid use is below limit so power available (for load shift)
            FR_PTU.loc[i,'P_Available'] = congestion_limit - forecast.iloc[i]
    return FR_PTU
```

Here below is the code for the automated ranking:

```
import pandas as pd
import numpy as np

# import functions:
import Flexibility_FUNC as FR

pd.options.mode.chained_assignment = None


# Start the ranking process
o = 0  # Used to go through the Analysis dataframe
rank_index = 0 # Starts analysis with all the ranks accepted.
Rank_opt = False # Used to exit the While loop.
EMS_best  = False # If no rank is found satisfying the conditions then the EMS correction is the best option left.
rank_increment = 0.01 # How much the rank increases at each iteration

# Initialize dataframes

# Final_Psh2 is the energy program after the rank analysis
Final_Psh2 = pd.DataFrame(index = AGR_E_prog.index, columns=['program','local'])
Final_Psh2.program = AGR_E_prog.program
Final_Psh2.local = AGR_E_prog.local

# temporary dataframe used to store info
Final_Psh_temp = pd.DataFrame(index = AGR_E_prog.index, columns=['program','local'])
Final_Psh_temp.program = AGR_E_prog.program
Final_Psh_temp.local = AGR_E_prog.local


Analysis = pd.DataFrame() # Contains all the analysis results for each ranks
Analysis_opt = pd.DataFrame() # Contains the analysis results of the optimal rank


while (Rank_opt == False):
```

```
35
36      rank_index = rank_index + rank_increment
37
38      #Bids are now filtered based on the rank
39      for i in range(0, len(rank.index)): # Energy program is filtered based on the rank
40          if (rank.loc[i] < rank_index).item():
41              Final_Psh_temp.program.iloc[i]  = EMS.external.loc[i]
42              Final_Psh_temp.local.iloc[i]= EMS.load_forecast.loc[i]
43
44      # Calculates the difference between the ranked Final E program and the AGR E program (
            checks the difference in E)
45      PshFlex['TotFlexUnservedRank'], PshFlex['TotFlexServedRank'] = FR.Difference_calc(
            Final_Psh_temp.program, AGR_E_prog.program, PshFlex.TotFlex)
46
47      # Caluclates the revenue and loss between the unranked and ranked E program (the function
             Fiancial_calc is applied)
48      temp = FR.Rank_analysis(PshFlex,price,penalty)
49      temp = (temp.reset_index()).loc[:,0]
50
51      # Conditions for which the optimal rank is accepted.
52      ratio_Psh = PshFlex['TotFlexServedUnrank'].cumsum().max()/PshFlex['TotFlexServedRank'].
            cumsum().max()
53      ratio_rev = temp.iloc[1]/temp.iloc[3]
54
55      for j in range(0,len(temp.index)): # Stores Analysis result into the Analysis dataframe
56          if (rank_index <= 1):
57              Analysis.loc[o,j] = temp.iloc[j]
58              Analysis.loc[o,0] = PshFlex['TotFlexServedUnrank'].cumsum().max()
59              Analysis.loc[o,2] = PshFlex['TotFlexServedRank'].cumsum().max()
60              Analysis.loc[o,'ratio_rev'] = ratio_rev
61              Analysis.loc[o,'ratio_Psh'] = ratio_Psh
62      # #checks whether the best ranking scenario has been reached or if more can be done
63      if (Analysis.ratio_rev.iloc[o-1] <= 1) and (Analysis.ratio_Psh.iloc[o-1] >= 1):
64          rank_index_opt = o*rank_increment # converts the o step into what rank it represents
65          Analysis_opt =  Analysis.iloc[o-1,:] # Saves the optimal rank into a seperate
                dataframe
66          Rank_opt = True # exits while loop
67      elif (rank_index >= 1): # all ranks have been tried no need to any further iteration
68          EMS_best  = True
69          Analysis_opt = Analysis.iloc[o-1,:]
70          Rank_opt = True
71      o = o + 1 #if none of the conditions are met then next iteration is done
72
73
74
75  #Rewrite the dataframe to make it more comprehensible
76  Analysis_opt['TotFlexServed_Unrank'] =  Analysis_opt.iloc[0]
77  Analysis_opt['TotFlexServed_Rank'] =  Analysis_opt.iloc[1]
78  Analysis_opt['Revenue_Unrank'] =  Analysis_opt.iloc[4]
79  Analysis_opt['Revenue_Rank'] =  Analysis_opt.iloc[5]
80  Analysis_opt['Ratio_Revenue'] =  Analysis_opt.iloc[2]
81  Analysis_opt['Ratio_Psh'] =  Analysis_opt.iloc[3]
82
83  if Analysis.index.max() == 0:
84      Analysis_opt['Optimal_Rank'] = 0
85  elif Analysis.index.max() != 0:
86      Analysis_opt['Optimal_Rank'] =  Analysis.index.max()*rank_increment
87
88  # Cleans the Analysis_opt dataframe
89  for i in range(1,7):
90      Analysis_opt = Analysis_opt.drop(Analysis_opt.index[0])
91
92  if EMS_best == False:
93      print("\n \n \n  RESULT OF RANK ANALYSIS: The most optimal index has been found. No more
            iteration is needed. \n \n \n ")
94  else:
95      print("\n \n \n RESULT OF RANK ANALYSIS: All rank indexes have been tried and none yield
            to an optimal result. \n \n \n ")
96
97  # Once the rank has been found, the energy program is filtered so that it can be proposed to
        the EMS.
```

```
if EMS_best  == False:
    for t in range(0,max(EMS.index)):
        if rank.iloc[t,0] < Analysis_opt.Optimal_Rank:
            Final_Psh2.program.iloc[t] = EMS.external.loc[t]
            Final_Psh2.local.iloc[t] = EMS.load_forecast.loc[t]
elif EMS_best  == True:
    for t in range(0,max(EMS.index)):
        Final_Psh2.program.iloc[i] = EMS.external.loc[i]
        Final_Psh2.local.iloc[i] = EMS.load_forecast.loc[i]
```