



Funded by
the European Union

Horizon Europe

EUROPEAN COMMISSION

European Climate, Infrastructure and Environment Executive Agency (CINEA)

Grant agreement no. 101160684

**U2DEMO**

Use of open-source P2P energy sharing platforms for energy Democratization

Deliverable D 4.1

P2P and Energy Sharing Design considering platform requirements

Document Details

Due date	30-11-2025
Actual delivery date	15-12-2025
Lead Contractor	Artelys
Version	1.0
Prepared by	Nicolas Fatras (Artelys), Salomé Aubri (Artelys), Rémi Simon (Artelys), Michaël Gabay (Artelys), Diogo Pinto Ferreira (INESC ID), Alena Kostalova (INESC ID), Larissa da Silva Montefusco (INESC ID), Hugo Morais (INESC ID), Wicak Ananduta (VITO), Yucun Lu (KU Leuven), Gonçalo Glória (R&D Nester)
Teams involved	Francisco Ferreira Reis (EDP NEW), Roberta Alonzo (EnGreen), Aliene van der Veen (TNO), Miguel Carvalho (Watt-IS), Nitin Gavhane (EWF)

Reviewed by	Paulo Pereira (INESC ID), Anibal Sanjab (VITO) & Glenn Reynders (KU Leuven)
Dissemination Level	Public

Project Contractual Details

Project Title	Use of open-source P2P energy sharing platforms for energy democratization
Project Acronym	U2Demo
Grant Agreement No.	101160684
Project Start Date	01-09-2024
Project End Date	29-02-2028
Duration	42 months

Document History

Version	Date	Contributor(s)	Description
0.1	31/07/2025	Nicolas Fatras et al.	First draft
0.2	17/11/2025	Anibal Sanjab, Paulo Pereira and Glenn Reynders	Internal review
1.0	12/12/2025	Nicolas Fatras	Review comments addressed and document finalized
1.1	17/12/2025	Nicolas Fatras	Editing reviews

Disclaimer

This document has been produced in the context of the U2Demo¹ project. Views and opinions expressed in this document are however those of the authors only and do not necessarily reflect those of the European Union or the European Climate, Infrastructure and Environment Executive Agency (CINEA). Neither the European Union nor the granting authority can be held responsible for them.

Acknowledgment

This document is a deliverable of U2Demo project. U2Demo has received funding from the European Union's Horizon Europe programme under grant agreement no. 101160684.



**Funded by
the European Union**

¹ <https://u2demo.eu/>

Executive Summary

Deliverable 4.1 sets the specifications, ontology and development framework for the energy sharing algorithms to be developed and integrated to the U2Demo platform in the subsequent tasks for Work Package 4.

The algorithms to be developed are identified based on the review and conceptualisation work done in previous work packages. The identified algorithms aim to be as reusable as possible, to make the work in U2Demo replicable in other contexts, while being suitable for the case-specific contexts of the U2Demo demo sites. The algorithms in WP4 are deduced from the proof-of-concept models presented in WP2, which cover the conceptual and mathematical formulation. The task starts by identifying commonalities across these different models, making these algorithms as reusable as possible. The suitability of the identified algorithms in the U2Demo context is then verified by aligning them with the site-specific activities expressed by community members, as reported in the surveys of community members done in WP1.

Four main types of algorithms emerge from this analysis:

- Centralised energy management system algorithms;
- Centralised market clearing algorithms (auction-based algorithms);
- Pricing mechanism algorithms;
- Heuristic benefit allocation algorithms.

The first two types of algorithms represent two different approaches to schedule (ex-ante) / settle (ex-post) assets within a community: the centralized energy management system receives constraints and costs from members or assets and schedules their dispatch on their behalf, while a centralised market clearing lets members or assets decide on an optimal dispatch schedule and validates or rejects this schedule in order to optimise the outcome for the community as a whole. Each of these paradigms can then be implemented in different contexts, at different geographical scales (household vs community optimisation) or different times (pre-dispatch planning vs post-dispatch settlement).

On the other hand, the pricing mechanism algorithms do not directly determine dispatch schedules of assets but instead try to influence the scheduling decisions of members, and thereby indirectly contribute to benefits at a community level.

Finally, heuristic benefit allocation algorithms do not involve any scheduling at all but rather look at redistribution methods for benefits acquired by the community. The very different structure and logic of these different algorithm types is largely influenced by the context in which they are implemented (Who takes decisions within the community? When are decisions taken? For what purpose are these energy sharing decisions being made?).

The identified algorithms are then standardised to ensure interoperability between algorithms and with the U2Demo platform. While this report does not detail the specific logic implemented within each algorithm, which is conceptualized and mathematically defined/modelled in WP2 and implemented in later tasks of WP4, it focuses on defining the data formats required to run

these algorithms on the U2Demo platform and interact with them. Therefore, detailed data transfer objects (DTO) based on classes with attributes and corresponding data types are defined for each algorithm (and sub-algorithm) type. The classes are defined following the object-oriented programming principle of inheritance, providing a high degree of modularity to the developed algorithms. This allows to quickly adapt the algorithm to different contexts while keeping the same fundamental building blocks. For the economic dispatch algorithms, the idea is pushed further by pre-defining constraints and cost functions of generic assets which simply need to be instantiated when defining a specific optimisation problem.

The corresponding DTO files are expressed using the Python Pydantic library and converted to OpenAPI standard json files, html files, mermaid files, and inheritance graphs, which can be found on the project's github page: <https://github.com/U2DemoProject/AlgorithmDTOsWP4T1>

Finally, a technical architecture for the communication protocol between these algorithms and the U2Demo platform is defined in the final section of this work, by specifying the containerisation approach, the message broker setup, and the message orchestration.

In summary, the formalism developed in this report helps to standardise the data communication:

- Between algorithms, to ensure that the different parties developing the algorithms have interoperable data formats. This is particularly important for example in energy sharing models where members run individual decision-making algorithms and the community runs algorithms for energy sharing either before (internal pricing models) or after (market clearing algorithm);
- Between each algorithm and the U2Demo platform, to ensure the data is received from the platform and sent back to the platform in the right format, using the right communication protocol.

The modelling standardisation provided in this report provides a foundation for a harmonised development of algorithms throughout the rest of WP4, for a successful integration with the U2Demo platform in WP3, and for real-life implementations of the case studies in WP5.

Table of Contents

Executive Summary	4
Table of Contents	6
List of Figures.....	8
List of Tables	10
Acronyms	11
1 Introduction	12
1.1 Scope and Objectives	12
1.2 Structure	12
1.3 Relationship with other deliverables	14
2 Model overview	15
2.1 Capacity, Energy and Revenue Allocation Models	15
2.1.1 Decentralized scheduling	15
2.1.2 Centralised scheduling	16
2.2 Internal energy pricing models	17
2.3 Auction-based local energy trading models	18
2.4 Pool-based Peer-to-Peer matching energy trading models.....	18
3 Translation of WP1 and WP2 into higher TRL algorithms	20
3.1 Conceptual framework	20
3.2 Identification of reusable algorithms based on models in WP2	21
3.3 Suitability of identified algorithms for U2Demo demo sites	27
3.3.1 Desired energy community activities within U2Demo	27
3.3.2 Demo sites use cases	31
4 Specifications for algorithm implementations	37
4.1 Common notation.....	37
4.1.1 Time parameters	37
4.1.2 General assumptions and hypotheses	38
4.1.3 Common library for all algorithms	40
4.2 Algorithm 1: Centralised energy management system.....	42
4.2.1 Common library for centralised energy management system algorithms	44
4.2.2 Algorithm 1-1: Centralised energy management system for Community Energy Management System	57
4.2.3 Algorithm 1-2: Centralised energy management system with “Peer-to-peer” preferences	59
4.2.4 Algorithm 1-3: Collective benefit allocation (settlement)	62
4.2.5 Algorithm 1-4: Individual portfolio optimisation	64
4.3 Algorithm 2: Centralised market clearing	65
4.3.1 General formulation and hypotheses.....	65
4.3.2 Time horizon	66
4.3.3 Sets and indices.....	66
4.3.4 Input data	67
4.3.5 Objective	71
4.3.6 Outputs	71
4.4 Algorithm 3: Pricing mechanism	72
4.4.1 General assumptions	72
4.4.2 Pricing model of the community manager.....	72
4.5 Algorithm 4: Heuristic benefit allocation.....	73
4.5.1 Solving methods.....	73

4.5.2	Input/ Output	74
5	Data schemes	75
6	Technical architecture	85
6.1	Overview	85
6.2	Deployment and Configuration	85
6.3	Communication Architecture	85
6.4	Message format	87
6.4.1	Input Data Structure	87
6.4.2	Output & Results	87
6.5	Real-Time Logging and Debugging	87
7	Conclusions.....	88
7.1	Summary.....	88
7.2	Main Challenges	89
7.3	Next deliverables.....	89
8	References	91
	APPENDIX A: Algorithm 1-1 – Multi-objective example	92
	APPENDIX B – Data Transfer Objects for Algorithms	96
B.1	Commons DTO	96
B.2	Commons energy management system DTO	99
B.3	Algorithm 1-1: Centralised energy management system for Community Energy Management System	106
B.4	Algorithm 1-2: Centralised energy management system with P2P preferences	108
B.5	Algorithm 1-3: Centralised energy management system for benefit allocation	111
B.6	Algorithm 1-4 Centralised energy management system – Individual portfolio optimisation DTO	114
B.7	Algorithm 2: Centralised market clearing	116
B.8	Algorithm 3: Pricing mechanism	120
B.9	Algorithm 4: Heuristic benefit allocation	122

List of Figures

Figure 1: Structure of Deliverable T4.1	13
Figure 2: Interaction of Deliverable 4.1 with other Deliverables	14
Figure 3: Schematic representation of Capacity, Energy and Revenue Allocation model with decentralised scheduling [1]	16
Figure 4: Schematic representation of Capacity, Energy and Revenue Allocation model with centralised scheduling [1]	17
Figure 5: Schematic representation of Internal energy pricing model [1]	17
Figure 6: Schematic representation of Auction-based local energy trading models [1].	18
Figure 7: Schematic representation of pool-based P2P matching energy trading model [1].	19
Figure 8: Conceptual distinction between models and algorithms	21
Figure 9: Sequence of algorithms for decentralised <i>Capacity, energy, and revenue allocation</i>	22
Figure 10: Sequence of algorithms for centralised <i>Capacity, energy, and revenue allocation</i>	23
Figure 11: Sequence of algorithms for <i>Internal energy pricing models</i>	23
Figure 12: Sequence of algorithms for centralised market clearing models	24
Figure 13: Sequence of algorithms for P2P matching	24
Figure 14: Mapping between models and algorithms	26
Figure 15: Linking EC activities to U2Demo algorithms	30
Figure 16: Link between demo site use cases and developed algorithms in WP4	35
Figure 17: Time parameters used in algorithms	38
Figure 18: Asset classification and household class representation	40
Figure 19: Block representation of Algorithm 1.1 (Decentralised vs Centralised versions)	43
Figure 20: Block representation of Algorithm 1.2 and 1.4	43
Figure 21: DTO for classes in Commons file	76
Figure 22: DTO for classes in Commons centralised energy management system	77
Figure 23: DTO for classes in Algorithm 1-1 (Centralised energy management system for Community Energy Management System)	78
Figure 24: DTO for classes in Algorithm 1-2 (Centralised energy management system with P2P preferences)	79
Figure 25: DTO for classes in Algorithm 1-3 (Centralised energy management system for Collective benefit allocation (settlement)	80
Figure 26: Centralised Energy Management System for Individual Portfolio Optimisation	81
Figure 27: DTO for classes in Algorithm 2 (Centralised market clearing)	82
Figure 28: DTO of classes for Algorithm 3: Pricing mechanism	83
Figure 29: DTO of classes for Algorithm 4: Heuristic benefit allocation	84

Figure 30: Communication architecture between the optimisation engine and U2Demo orchestrator86

List of Tables

Table 1: Mapping of EC activities identified in WP1 with algorithms developed in WP4	27
Table 2: Use case descriptions for demo sites	32
Table 3: Parameters used for common notation across algorithms	37
Table 4: Contract class parameters	40
Table 5: Meter class parameters	41
Table 6: Supplier class parameters	41
Table 7: Community member class parameters	41
Table 8: Household class parameters	42
Table 9: Energy Community Manager parameters	42
Table 10: Notation for sets and indices in the common library	44
Table 11: Parameters used for battery modelling	46
Table 12: Parameters used for EV modelling	47
Table 13: Parameters used for EV charger modelling	48
Table 14: Parameters used for PV modelling	48
Table 15: Heat pump parameters	49
Table 16: Parameters used for flexible load modelling	49
Table 17: Parameters used for unflexible load modelling	50
Table 18: Parameters used to ensure feasibility of the optimisation problem	50
Table 19: Sets and indices used for Centralised energy management with P2P preferences	59
Table 20: Attributes used for modelling the quadratic cost function of peers	60
Table 21: Attributes used for expressing the product differentiation parcel in the OF	60
Table 22: Attributes for modelling peers in the energy management system problem with P2P preferences	61
Table 23: Sets and indices used for the centralised market clearing algorithm	66
Table 24: Attributes of the OrderStep class	67
Table 25: Attributes of the BlockOrder class	68
Table 26: Attributes of the TimestepOrder class	69
Table 27: Attributes of the Order class	69
Table 28: Attributes of the ObjectiveRanking class	70
Table 29: Attributes used for ClearingPriceModel implementing a marginal price model	70
Table 30: Attributes of the PricingMechanism class used to calculate internal prices	72
Table 31: OF correlation with β factor	95

Acronyms

AMQP	Advanced Message Queuing Protocol
BESS	Battery Energy Storage System
BRP	Balance Responsible Party
BS	Battery Storage
CEMS	Community Energy Management System
CM	Community Manager
DER	Distributed Energy Resource
DTO	Data Transfer Object
EC	Energy Community
EV	Electric Vehicle
GO-PACS	Congestion Management Platform of the Dutch Grid Operators
HEMS	Home Energy Management System
HVAC	Heating, Ventilation, and Air Conditioning
ID	Identifier
OF	Objective Function
P2P	Peer-to-Peer
PV	Photovoltaic
SoC	State of Charge
WP	Work Package

1 Introduction

1.1 Scope and Objectives

The aim of Deliverable D4.1 is to describe the algorithms that will be developed in WP4, by combining the functional model designs proposed in WP2 with the technical specifications and requirements of the U2Demo platform developed in WP3.

The U2Demo platform works as the interface through which all information required for the energy community (EC) to function is exchanged. To successfully implement these information flows through the platform, the data schemes, environment requirements and the technical architecture must be defined. These specifications are, in turn, dependent on the functionalities of the algorithms which are being implemented. This section will therefore also outline the detailed design structure of the algorithms developed in U2Demo. While the details of the algorithm implementation are left to subsequent tasks (i.e. T4.3 and T4.4), the algorithm specifications described in this deliverable ensure the algorithms follow a general structure, which is compatible with the U2Demo platform.

The output of Deliverable D4.1 sets the foundation for the developments in T4.3 “Peer-to-peer trading algorithms” by providing a formalism which allows for harmonising and extending the implementation of the models developed in Deliverable D2.3 [1]. This excludes monitoring and forecasting services at an individual household level, which are covered in T4.2, and the external flexibility services, which are covered in T4.4. However, as these algorithms are interdependent and communicate with each other, the data formalism defined in this deliverable is important for all subsequent tasks in WP4.

1.2 Structure

The first part focuses on identifying relevant algorithms to implement based on the work done in previous work packages:

- Section 2 focuses on analysing and summarising the models developed in WP2, to outline the different logical steps within each model and identify potential commonalities.
- Section 3 determines the relevant algorithms to develop based on the commonalities identified in the first part, and cross-validates the suitability of the proposed algorithms with the business cases of the U2Demo demo sites described in WP1.

The second part focuses on providing standardised specifications for each algorithm:

- Section 4 specifies the algorithms’ inputs, outputs, and logical structure, in order to standardise the communication protocol with these algorithms.
- Section 5 provides the data schemes for each algorithm in the form of (DTOs)

The third part focuses on describing the technical architecture implemented around these algorithms:

- Section 6 outlines the technical architecture recommended to integrate the different algorithms with the U2Demo platform developed in WP3.

An overview of the different parts and the sections contained in each part is provided in Figure 1.

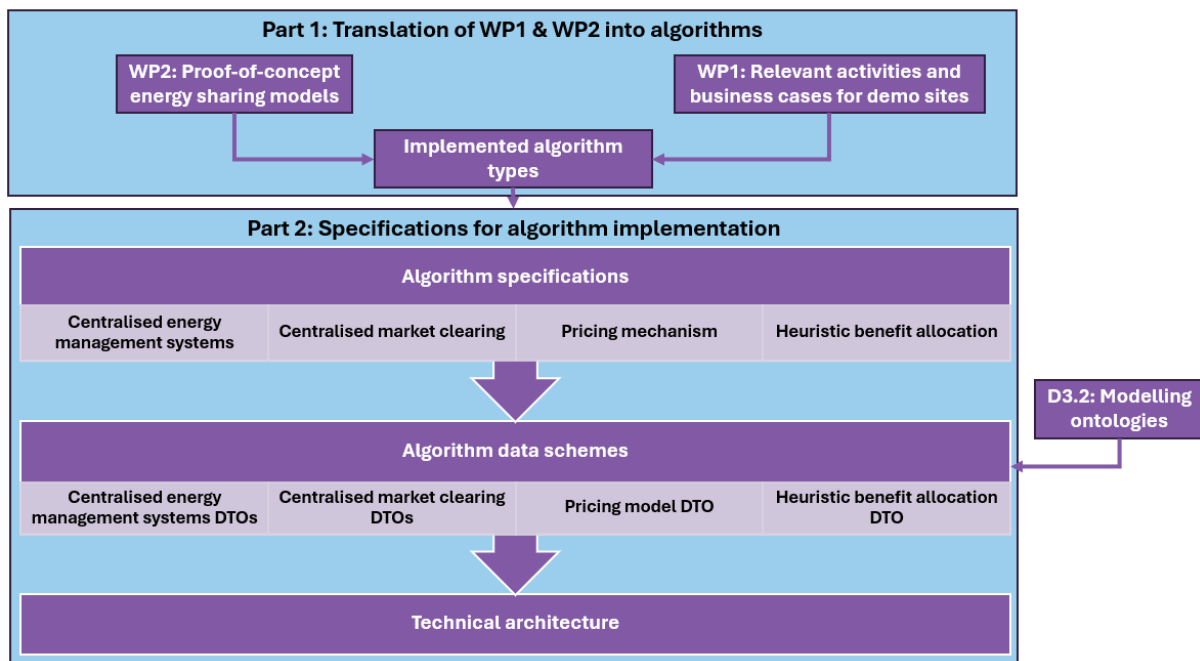


Figure 1: Structure of Deliverable T4.1

1.3 Relationship with other deliverables

The interaction of T4.1 with other deliverables is illustrated below (Figure 2).

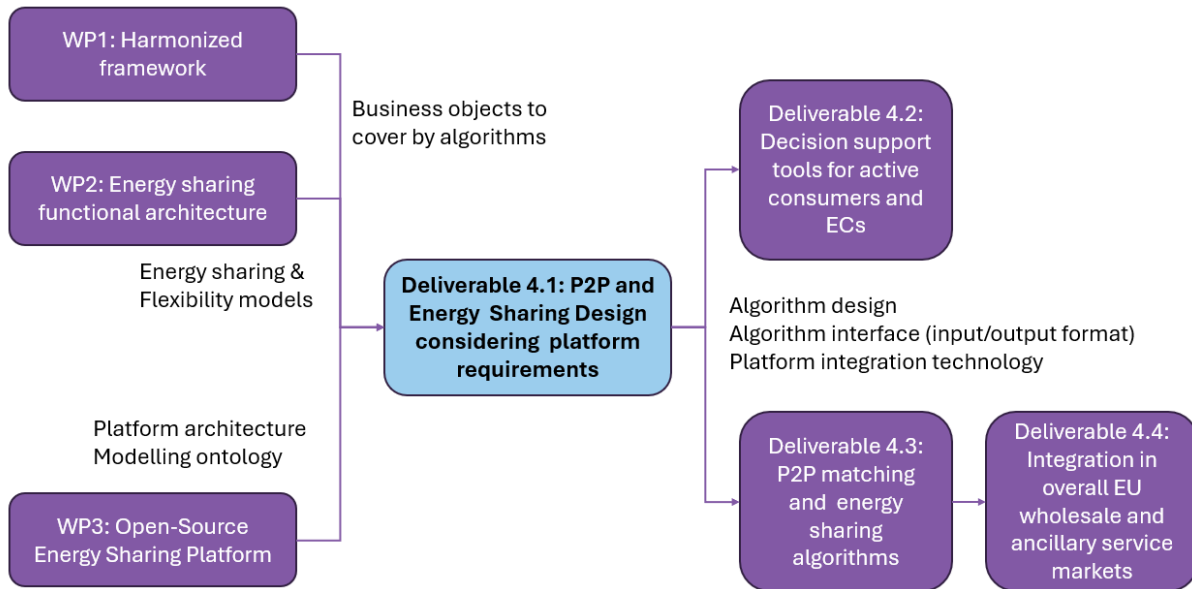


Figure 2: Interaction of Deliverable 4.1 with other Deliverables

2 Model overview

In Task T2.3, four main business models for energy communities have been identified. These models are presented below, using the naming convention introduced in Deliverable D2.3 [1]. The below is a short replication/summary of the main models developed in WP2 and initially reported in D2.3. The reader is referred to D2.3 for the detailed elaboration of the models, their motivation and use.

2.1 Capacity, Energy and Revenue Allocation Models

Energy and Revenue Allocation Models optimise energy sharing at the community level, with or without collective investment among community members and how the revenue from the energy sharing activity is distributed among the members, as presented in [1]. Two versions of this model are examined, which differ in the degree of decentralisation in energy management and information sharing.

2.1.1 Decentralized scheduling

In the decentralised scheduling scenario, energy management is first done at the individual level but exchanges between members are coordinated centrally at the community level, as presented in [1]. First, each community member is building its own individual portfolio strategy where it determines the optimal operation of their local assets, according to personal objectives. The outputs of this process are individual energy injection and offtake profiles that are sent to the community manager (CM). In a second phase, the CM collects the individual profiles and the technical parameters of the collective assets to serve as inputs of the centralised optimisation process, which optimises the energy flows inside the community and with the external grid. Then, the benefits generated from both the individual and collective assets are allocated among members. A simplified diagram for Model 2.1 is presented in Figure 3.

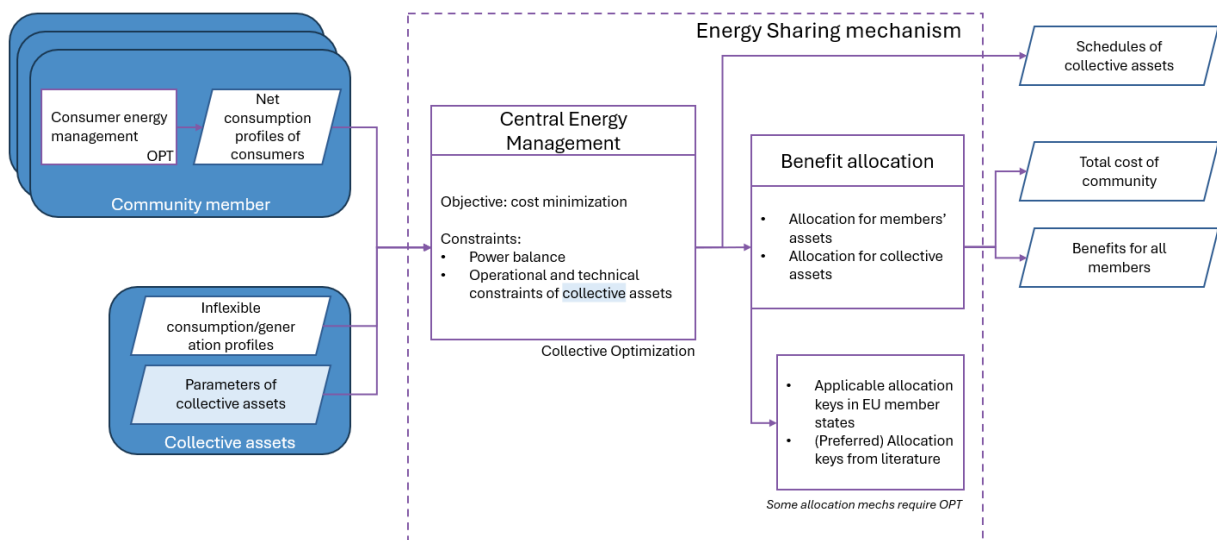


Figure 3: Schematic representation of Capacity, Energy and Revenue Allocation model with decentralised scheduling [1]

2.1.2 Centralised scheduling

The centralised scheduling scenario, the same overall goal and structure as the decentralised version are shared, but it presents a higher level of centralization, as presented in [1]. In this variant, community members no longer perform individual portfolio optimisation. Instead, they directly provide the parameters of their assets to the CM, who conducts a centralized optimisation process to jointly schedule both individual and collective assets. The final phase remains identical to the decentralised version, with the benefit allocation redistributing the resulting gains among members. A simplified diagram of the centralised scheduling version is presented in Figure 4.

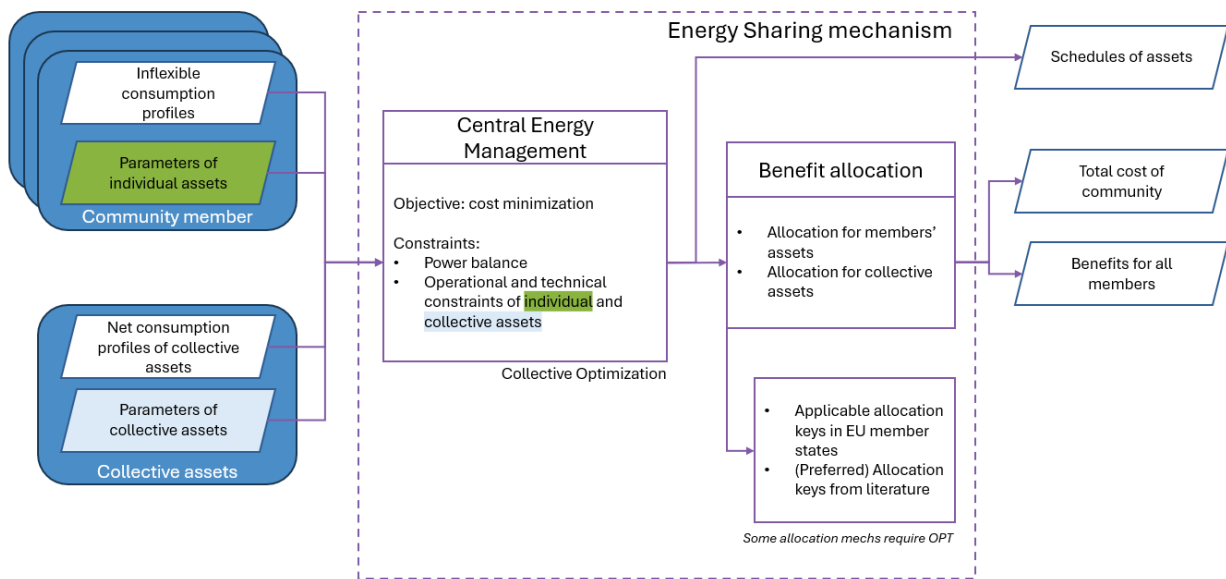


Figure 4: Schematic representation of Capacity, Energy and Revenue Allocation model with centralised scheduling [1].

2.2 Internal energy pricing models

Internal energy pricing models represent the pricing mechanism developed within an EC and follow the steps illustrated in Figure 5 [1]. In the first step, the CM gets the forecasted aggregated consumption and local generation profiles of the community. The CM applies a pricing mechanism to determine the internal energy prices within the community. Households take into account the price signal sent by the CM to individually construct their own portfolio strategy. After the real-time operation phase, the CM revises the internal prices based on the realised aggregated consumption and generation profiles. The same pricing mechanism is applied. This final computation holds the definitive internal prices within the community.

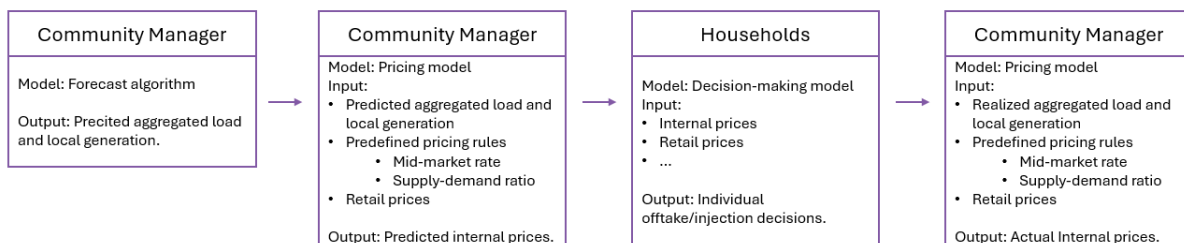


Figure 5: Schematic representation of Internal energy pricing model [1].

2.3 Auction-based local energy trading models

Auction-based local energy trading models are designed to facilitate energy trading within the community, as presented in [1].

Each community member submit offers and/ or bids based on their individual bidding strategy (Figure 6) [1]. The CM collects the members' submission and centrally clears the market. An equilibrium price and the energy allocation among participants are determined. Residual demand or supply is settled with the external grid at the retail price. Finally, costs and revenues are distributed among members according to the traded quantities, market outcomes and the settlement rule applied.

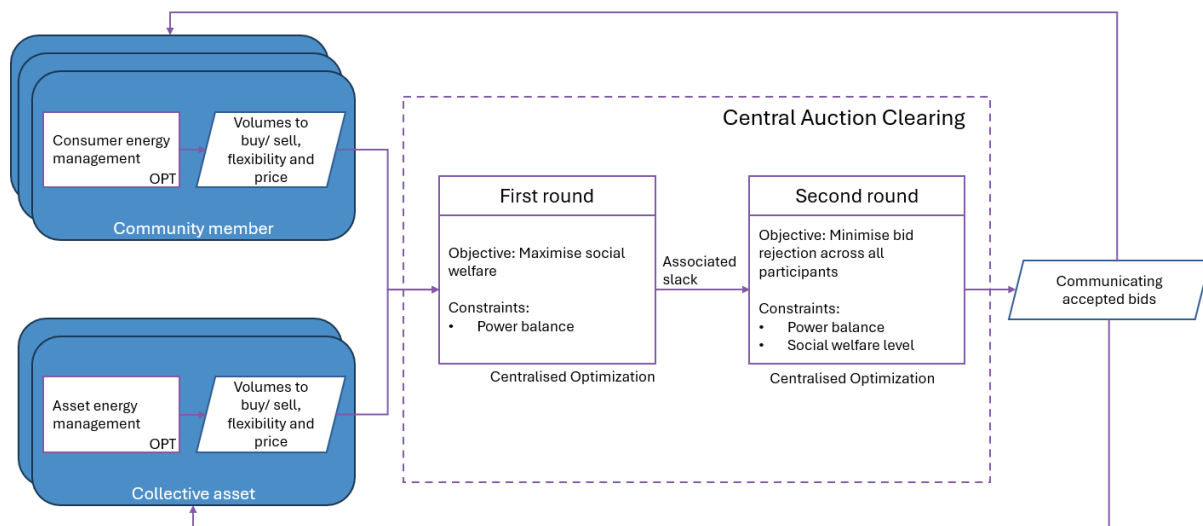


Figure 6: Schematic representation of Auction-based local energy trading models [1].

2.4 Pool-based Peer-to-Peer matching energy trading models

The pool-based P2P (Peer-to-Peer) matching energy trading models represent a centralised P2P trading structure, as presented in Figure 7 [1]. Each peer specifies their internal production or consumption cost but also adds a product differentiation cost for each member within the community. This additional product differentiation allows members to express their preference to buy or sell from specific community members, rather than from anyone in the community in general.

The final production or consumption profile scheduled for each member is still determined by a central algorithm, which receives all the prices and operational constraints as inputs and aims to minimise the sum of operational costs and product differentiation costs. For this reason, this model is referred to as a centralised P2P model, rather than a bilaterally cleared P2P model.

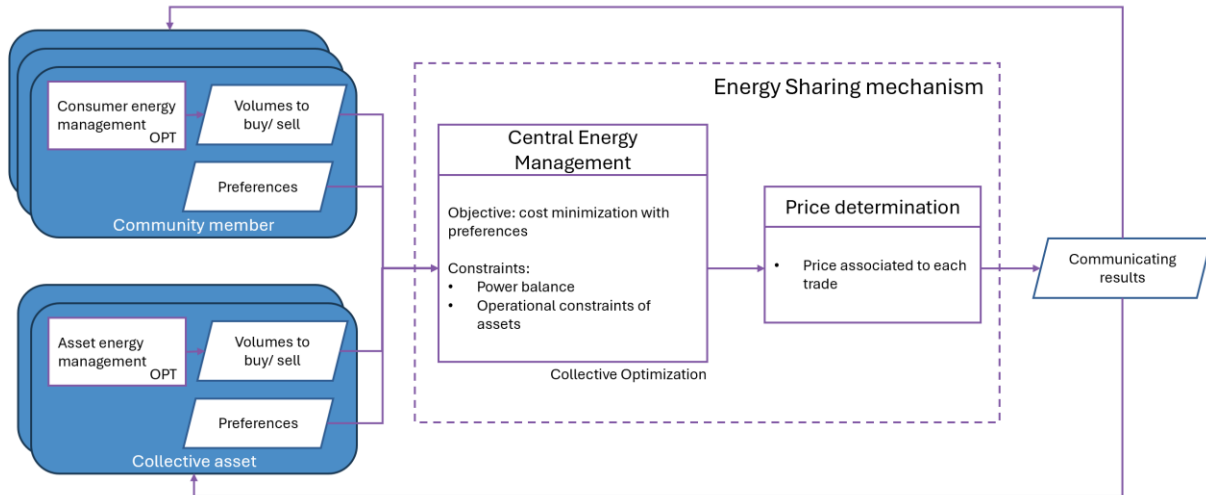


Figure 7: Schematic representation of pool-based P2P matching energy trading model [1].

3 Translation of WP1 and WP2 into higher TRL algorithms

3.1 Conceptual framework

The models from WP2 presented in the previous section have been developed in the context of the U2Demo project, whose scope is defined in WP1 through the identification of relevant activities and business cases. WP2 has developed different concepts of energy sharing and trading models, including their mathematical development and initial implementation. The models developed in WP2 present, hence, a proof-of-concept of different energy sharing/trading models/schemes. The implementation of these proofs of concept in the U2Demo platform in WP4 must achieve a higher technology readiness level (TRL) level, reaching TRL 8. To achieve such an increase in TRL, several criteria must be satisfied.

Reusability

The developments made in WP4 aim to be reusable, in order to provide solutions that can be reused outside of the U2Demo context, which is another important feature of high TRL-level solutions. In order to ensure reusability, a modular approach is chosen to implement and integrate the models in the U2Demo platform. Each model described above is disaggregated into different blocks, referred to hereafter as algorithms. For example, Model 3 is composed of a “Forecasting” algorithm used in step 1, a “Decision making” algorithm used in step 3, and a “Pricing Model” algorithm, used in steps 2 and 4. The same algorithm can thus be used several times within a model.

The same algorithm can also be used across different models. Taking the “Decision making” algorithm from the internal energy pricing model as an example, we see that it can also be applied for each member’s individual optimisation step in the decentralised scheduling scenario. More generally, we see that it is a centralised techno-economic dispatch problem. The same algorithm could therefore be used to solve the problem at a community level instead of a household level, by increasing the number of assets included in the optimisation problem and adapting the objective function (OF) to reflect community-level preferences rather than individual objectives.

By applying this reasoning to all the models described in section 2, we identify a set of algorithms to implement in WP4 to achieve energy sharing algorithms in energy communities. The aim is to find the minimum amount of algorithm categories which can cover all the models developed in WP2. The distinction between algorithms and models is illustrated in Figure 8.



Figure 8: Conceptual distinction between models and algorithms

Suitability in deployed environment

While aiming to be reusable, the developed algorithms should also be suited to the specific site conditions of the U2Demo context. Indeed, according to the TRL classification in [2], to achieve such a high TRL, the models must not only be integrated into an operational platform, they must also be tested in the target environment. This requires that the models can handle the data objects that are expected to be received from the demo site, at a suitable frequency. For example, models using minute-based active control are not suitable if the demo sites in which the models are supposed to run do not have any controllable actuators and only share the benefits of the EC ex-post. WP4 will therefore only develop models which can realistically run in the demo sites considered in the context of U2Demo. This requires filtering the proposed models to ensure they are aligned with the demo site business cases. For this purpose, the energy sharing activities are identified in WP1.1. and the demo site business cases identified in WP1.4 are analysed to ensure the algorithms identified based on the models in WP2 can cover all the relevant business cases and activities identified for the demo sites in WP1.

3.2 Identification of reusable algorithms based on models in WP2

Each model developed in WP2 has been subdivided into a sequence of steps, following the description from Section 2. These steps are each affiliated with an algorithm which will allow to solve the model. The same name is used for algorithms which are used across different models, to emphasise their reusability.

In the decentralised *Capacity, energy, and revenue allocation model*, each community member employs a local decision-making algorithm to generate an individual energy profile derived from their portfolio optimisation. The CM then applies a centralised dispatch algorithm to determine the optimal allocation of energy flows within the community and the schedule of the collectively invested assets. The final phase involves the benefit allocation process, which is carried out through a settlement algorithm. This settlement algorithm is implemented in two different ways:

- Heuristic approach: Uses predefined sharing rules, agreed upon by the community, to ensure a fair and transparent distribution of collective gains. This method takes into account members' physical generation and consumption profiles, as well as their individual grid injection and offtake prices;
- Cooperative game-theoretic approach: Determines the optimal allocation ex post, by evaluating the impact of each member on the community based on the results of subsets of the original problem called coalitions [1]. This requires an iterative approach on the centralised energy management system problem, considering different coalitions in each iteration.

The Figure 9 summarises the decentralised scheduling.

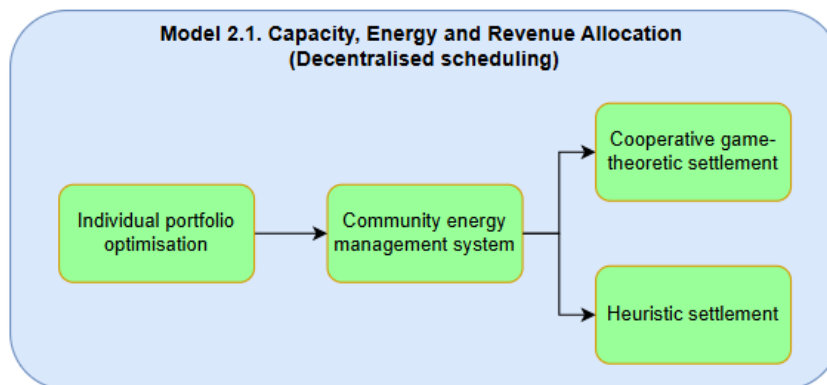


Figure 9: Sequence of algorithms for decentralised *Capacity, energy, and revenue allocation*

In the centralised scheduling approach (Figure 10), community members take on a passive role by transmitting the parameters of their individual assets directly to the CM. The individual portfolio optimisation step is therefore omitted in this configuration. The CM employs a centralised dispatch algorithm to optimise the operation and energy sharing of all assets, collective and individual, within the community. The subsequent benefit allocation phase is carried out using a settlement algorithm, identical in principle to that applied in the decentralised scheduling approach.

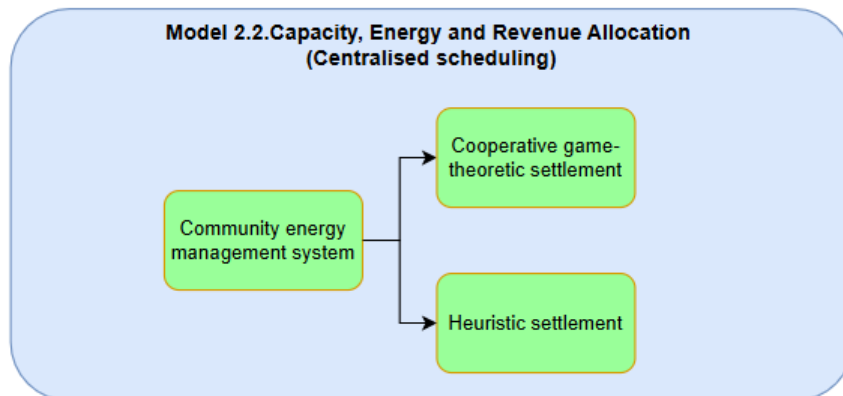


Figure 10: Sequence of algorithms for centralised *Capacity, energy, and revenue allocation*

In *Internal energy pricing models* (Figure 11), the energy sharing mechanism is structured into four main stages. In the first stage, the CM forecasts the aggregated consumption and local generation using the prediction algorithms developed in Deliverable D2.2 [3]. In the second stage, the manager applies the pricing model to determine the internal prices for the upcoming day and communicates these prices to community members, ideally one day in advance. In the third stage, each household independently optimises its energy usage through an individual portfolio optimisation process, taking into account the announced internal price signals. Finally, in the fourth stage, the CM recalculates the internal prices ex post, using the same pricing model but based on the realised consumption and generation profiles.

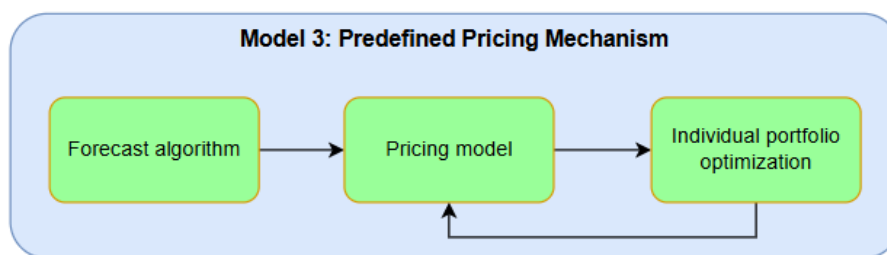


Figure 11: Sequence of algorithms for *Internal energy pricing models*

The *auction-based local energy trading mode* (Figure 12) is a sequence of three stages. The individual portfolio optimisation is first executed by each peer to determine the volume bid into the community, and at what price. The volume which a community member bids corresponds to the volume which it cannot cover internally (residual consumption or overproduction). The bid price corresponds to the marginal value of this volume, which is obtained by taking the cost of the marginal asset allowing to satisfy the energy balancing constraint in the individual portfolio optimisation problem.

The outputs of the individual portfolio optimisation algorithms run by each community member are the inputs of the centralised market clearing algorithm that is launched by the CM. Finally, a heuristic settlement algorithm is possibly applied to reallocate the resulting revenues and

costs between the community members. An example of heuristic settlement could be fixed sharing keys, as outlined in Deliverable D2.3 [1].

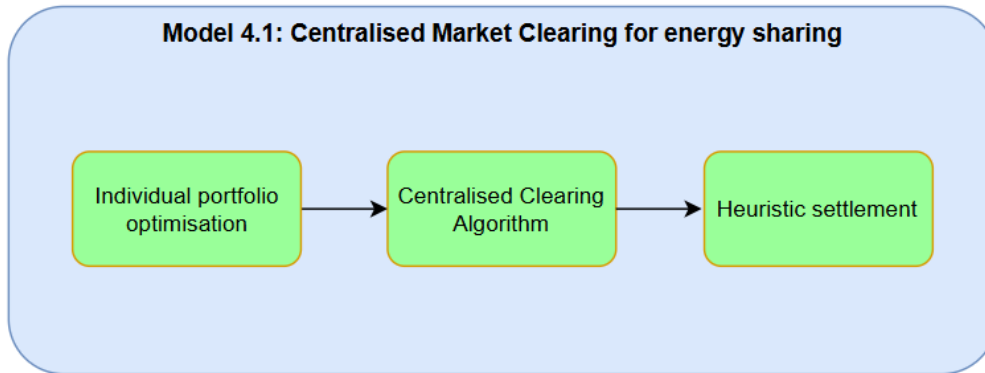


Figure 12: Sequence of algorithms for centralised market clearing models

The *Pool-based P2P matching energy trading model* (Figure 13) relies on a centralised energy management system algorithm, just as in the centralised version of the *Capacity, energy, and revenue allocation model*, to optimally allocate generation and consumption volumes within members of the community. However, participants do not provide operational constraints to the dispatch algorithm, but only the volumes which they are willing to exchange in the community. Those are the outputs of an individual optimisation problem they executed beforehand. The members then associate an internal cost function to this volume, as well as product differentiation costs, to represent their preference for some partner over others. The central optimisation problem allocates exchange volumes between peers based on the peer’s internal cost function and their stated product differentiation costs. The price of each bilateral volume exchange is defined by the stated internal and differentiation costs.

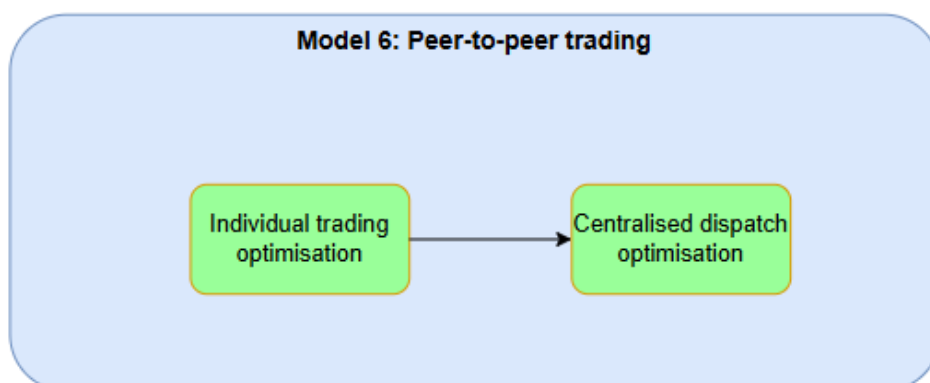


Figure 13: Sequence of algorithms for P2P matching

It is important to understand that the sequence of algorithms plays a crucial role in shaping the subsequent stages of the work. These interlinked processes define the required data

structures and interfaces between algorithms. Ensuring compatibility between algorithmic inputs and outputs is essential for smooth integration across the different steps. For example, in Model 2.1, the results produced by the individual portfolio optimisation must be formatted to align precisely with the input requirements of the centralised dispatch stage, thereby guaranteeing consistency and data flow throughout the entire framework.

From the preceding analysis, four algorithms stand out as central to the energy sharing and allocation mechanisms:

- the centralised energy management system algorithm, whether at individual household, sub-coalition, or community level. Bilateral P2P energy sharing is also included in this category, as the trading decision is decided centrally by the algorithm, and the bilateral nature is only expressed as preference weights in the OF;
- the centralised market-clearing algorithms;
- the internal pricing models;
- the heuristic settlement algorithms.

These constitute the core responsibilities of the CM, forming the backbone of the operational framework and determining the efficiency of energy allocation.

This framework is summarised in a mapping between the theoretical models and their corresponding algorithms, as illustrated in Figure 14.

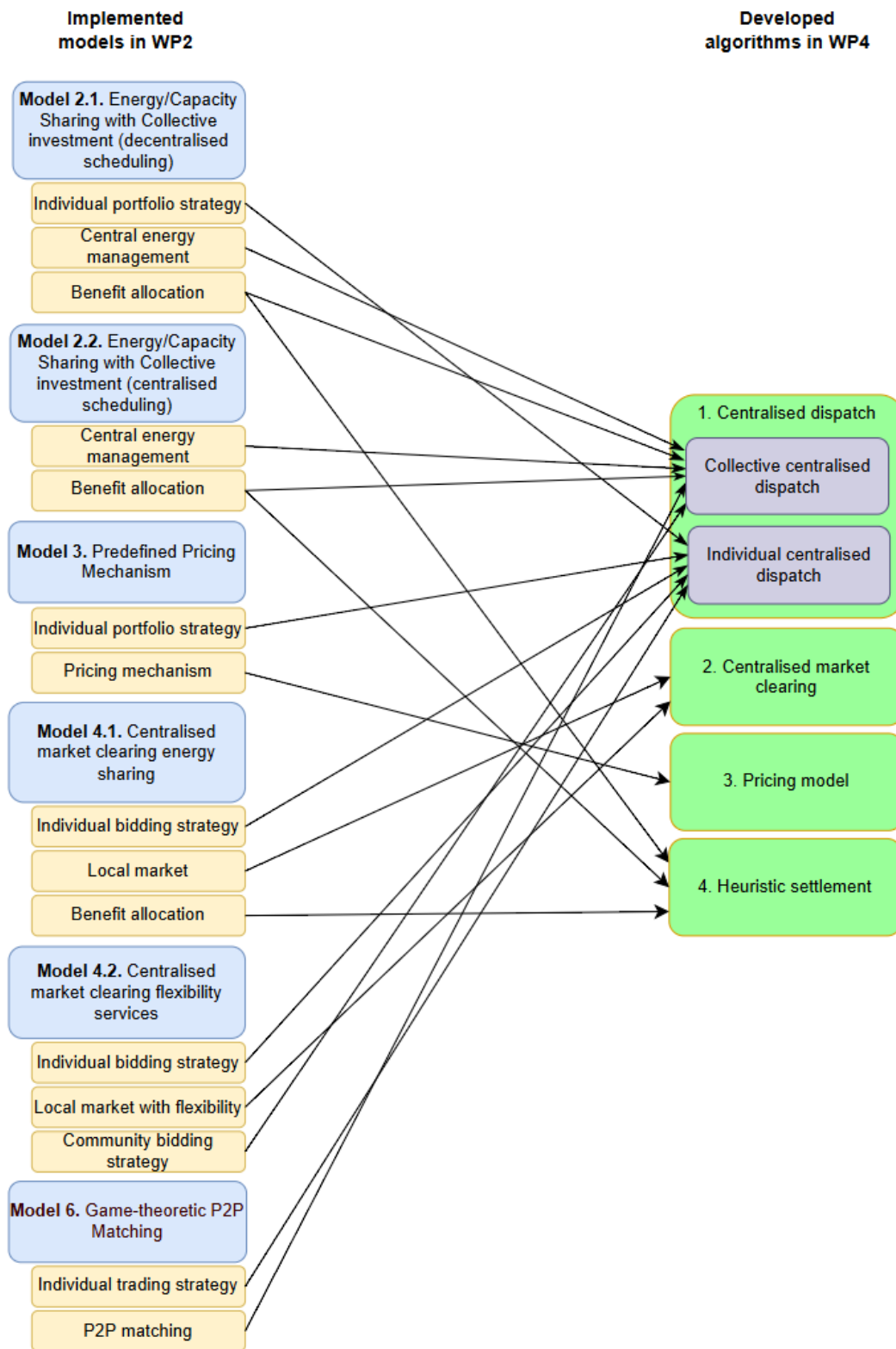


Figure 14: Mapping between models and algorithms

3.3 Suitability of identified algorithms for U2Demo demo sites

3.3.1 Desired energy community activities within U2Demo

In Deliverable D1.2 of the U2Demo project [4], community members across the four demo sites (Belgium, the Netherlands, Italy, and Portugal) were interviewed in order to better understand the activities they wish to see implemented within their respective communities.

The algorithms developed in the U2Demo project should enable these activities. The different activities are therefore listed below with a brief explanation, in order to conclude which algorithm can address this activity (Table 1). Doing so ensures that the set of algorithms proposed covers a large enough spectrum of desired activities.

Table 1: Mapping of EC activities identified in WP1 with algorithms developed in WP4

Activity	Description	Corresponding algorithm
Joint self sub-supply	The community is an alternative energy supplier for the members, which also have their own supplier. Community members therefore have to solve an individual optimisation problem to make an arbitrage between community prices and supplier prices. The community prices can be set either by an internal pricing mechanism or based on the community prices which can be expected from a centralised market clearing approach.	- Individual optimisation + internal pricing mechanism - Individual optimisation + centralised market clearing
Joint self-supply	The community is the sole energy supplier of the members. Prices are therefore either decided internally and members schedule loads accordingly or the community prices are set based on bids offered by the members after internal optimisation at an individual level.	- Individual optimisation + internal pricing mechanism - Individual optimisation + centralised market clearing
Collective kWmax balancing	The community constrains the maximum amount extracted from the grid. This can be implemented either through direct control of the assets by the CM, which represents a centralised dispatch situation, or by	- Collective centralised dispatch - Internal pricing mechanism

	influencing the consumption pattern of community members through strong enough price signals.	
Collective self-consumption registration	Community members redistribute the benefits of the community ex-post in the settlement phase. This can be done either through rule-based calculations defined by the community in advance, such as weighted sharing coefficients, or it can be implemented using a collective centralised energy management system at the community level, in case quantitative metrics such as stability, efficiency, or fairness should be optimised.	<ul style="list-style-type: none"> - Heuristic settlement - collective centralised energy management system
Collective flex activation delivery to flex service provider	<p>The available flexibility in the community is aggregated and provided as a service outside of the community.</p> <p>This selected flexibility volumes can be chosen either through a centralised dispatch algorithm, if members provide the right to do so to the CM, or through a centralised market clearing system, if community members submit bids at a certain price which are cleared based on the flexibility demand expressed by the flexibility market.</p>	<ul style="list-style-type: none"> - Collective centralised energy management system - Centralised market clearing
Optimisation of local sustainability goals	This implies a central optimisation of operational dispatch based on the sustainability goals predefined by the community.	<ul style="list-style-type: none"> - Centralised market clearing
Energy monitoring and advice services	Monitoring community members' production and load profile and advising on their operational profile accordingly. This will be done in WP4.2 and is not covered in this deliverable, which focuses on energy sharing.	<ul style="list-style-type: none"> - Not relevant in this deliverable
Energy community as Esco	The Esco (Energy Service Company) optimises the community / community members' investments to minimise their operational costs (electricity bill). This optimisation is out of scope of this deliverable, as it considers long-term investment, rather	<ul style="list-style-type: none"> - Not covered in this deliverable.

	than optimisation of short-term energy sharing.	
Collective controlling rights	The CM has full control over the dispatched assets. These assets can be controlled by a centralised optimisation algorithm.	- Collective centralised dispatch
Collective community investment	The community decides how to invest in energy assets at the community level. This investment decision process is not modelled by energy sharing algorithms. However, the operational optimisation of these shared assets can be included in the algorithms presented in this deliverable.	- Not covered in this deliverable.
Secondary billing	The community members' original bills, based on their individual consumption and production profiles, is adjusted to redistribute the financial benefits of energy netting across community members. This redistribution can be done in different ways. Based on the feedback given by community members done in Deliverable D1.2 [4], this redistribution includes a heuristic approach through sharing coefficients predetermined by community members. In addition, more advanced redistribution methods such as worst-case value excess and the Shapley values are also implemented.	- Heuristic settlement
Collective self-balancing	A target profile is communicated to the Balance Responsible Party (BRP) to provide balancing services within the BRP's portfolio. Deviations from the target profile are penalised at the imbalance price. This prescheduling can be defined either a centralised dispatch algorithm, or through a centralised market clearing algorithm, which defines the target profiles based on the volumes cleared by each community member.	- Collective centralised dispatch - Centralised market clearing

The activities considered for the U2Demo sites are shown on the left part of Figure 15. Within these communities, activities involving physical energy exchange are shown in dark blue, while

the others are shown in light blue. The activities are then linked to the algorithms identified in this task which allow to implement these activities. Activities which are not covered by any algorithm are shown in yellow.

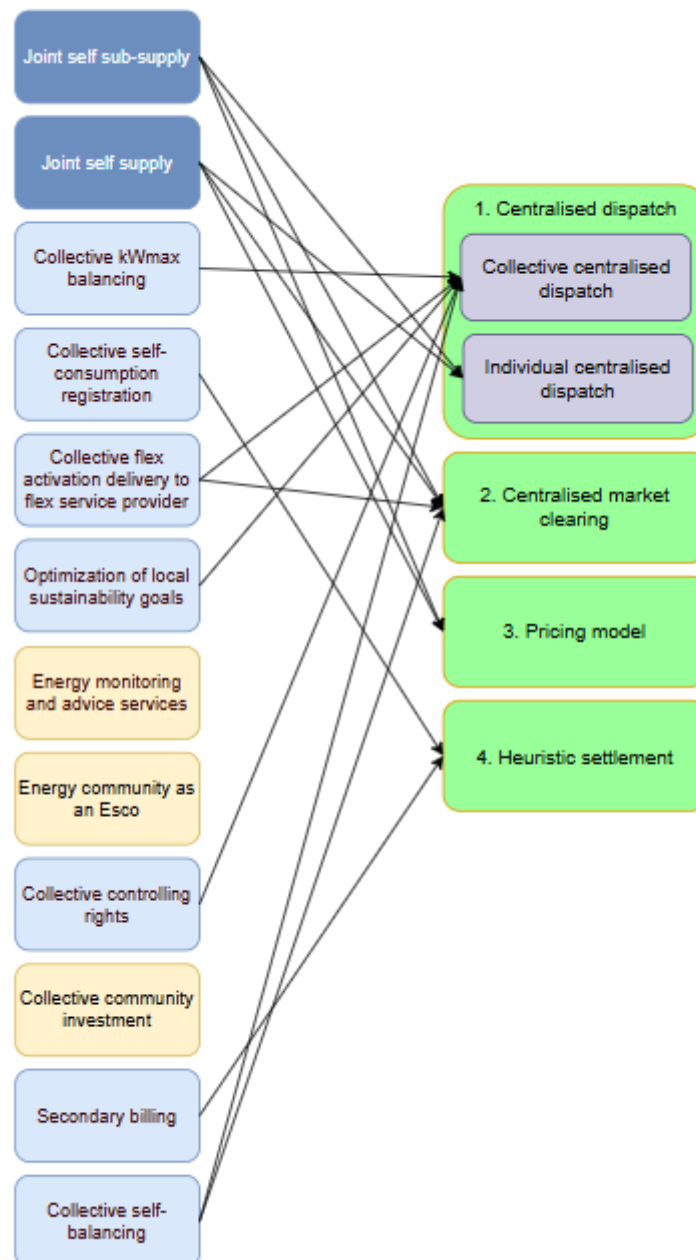


Figure 15: Linking EC activities to U2Demo algorithms

Figure 15 shows that the proposed algorithms allow to represent the majority of the EC activities identified in Deliverable D1.2 [4]. In particular, the energy sharing activities, which are

highlighted in navy blue in Figure 15 and are the focus of Deliverable D2.3 [1], are covered by a variety of algorithms. Additionally, the algorithms allow to model explicit flexibility services such as “*kWmax balancing*” and “*Collective flex activation delivery to flex service provider*”, which is the focus of T2.4. Finally, many of the activities which do not involve energy sharing but require collective decision making (such as “*Collective self-balancing*” or “*Collective controlling rights*”) can be implemented through Centralised dispatch algorithms. Investment- and monitoring-based activities are the only ones which cannot be covered by the algorithms, but as they are out of the energy sharing scope, this is deemed acceptable.

In general, it appears that implementing modular centralised dispatch algorithms is essential to address the different activities which demo sites expect from the U2Demo project.

3.3.2 Demo sites use cases

The analysis done in Deliverable D1.4 “U2Demo Use Case Repository” [5] identifies the use cases which the demo sites are hoping to implement through the U2Demo project. The algorithms hosted by the U2Demo platform should therefore be able to address each of these use cases.

As noted in D1.4 and throughout the conversations with the demo site representatives along the different General Assemblies of the U2Demo project, a distinction is made between present and future use cases. Present use cases can be implemented on the demo sites with the currently available infrastructure setup and community mindset, while future use cases are something the demo sites hope to achieve after further efforts in building up the EC, both through financial investment and after gaining more experience as an EC.

Table 2: Use case descriptions for demo sites

Demo site	Available assets	Level of control	Use case
Belgium	<ul style="list-style-type: none"> • PV • Batteries 	<p>Present:</p> <ul style="list-style-type: none"> • No active control <p>Future:</p> <ul style="list-style-type: none"> • Control by community manager through smart meters 	<p>Present</p> <ul style="list-style-type: none"> • Ex-post settlement based on sharing coefficients determined by the community. The community's sub-supplier (Klimaan) allocates the shared energy based on the sharing coefficients and the member's measured consumption / production profile.
			<p>Future</p> <ul style="list-style-type: none"> • Remote control of flexible assets (batteries) by community's sub-supplier (Klimaan). The assets are installed at the household level but owned and operator by Klimaan => centralised dispatch algorithm • Advice services to community members to minimise energy bills => assumes an optimal dispatch profile calculated based on a centralised dispatch algorithm.
Italy			<p>Present</p>

	<ul style="list-style-type: none"> • Heat pumps + boiler • EV • PV+Battery Energy Storage System (BESS) +Smart meter 	<ul style="list-style-type: none"> • Households can control assets individually. 	<ul style="list-style-type: none"> • Ex-post settlement (method undefined)
			<p>Future</p> <ul style="list-style-type: none"> • Centralised scheduling control of flexible assets => centralised economic dispatch
Netherlands	<p>Individual:</p> <ul style="list-style-type: none"> • PV • EV • Battery • Heatpump <p>Collective:</p> <ul style="list-style-type: none"> • Battery • Charging station 	<ul style="list-style-type: none"> • Community manager can control a collective battery • Individual shops can schedule their flexible assets based on community prices (no specification whether automated or manual scheduling) 	<p>Present</p> <ul style="list-style-type: none"> • kWmax contract with grid => no energy sharing optimisation implemented • External flexibility services to day-ahead congestion capacity market Congestion Management Platform of the Dutch Grid Operators (GO-PACS) based on forecasts => no energy sharing optimisation implemented • Internal price model based on day-ahead market prices + expected demand elasticity (not implemented yet) => heuristic pricing algorithm • Individual decision-making model for asset scheduling based on community prices => individual portfolio optimisation • Collective asset (battery) scheduling based on expected peaks => no energy sharing optimisation implemented



			<p>Future</p> <ul style="list-style-type: none"> • Use auction-based clearing (for settlement, for asset scheduling) => market clearing algorithm • External flexibility services to intraday congestion capacity market GO PACS => market clearing algorithm to select flexible volumes amongst community members.
<p>Portugal</p>	<ul style="list-style-type: none"> • PV • Battery 	<p>Present:</p> <ul style="list-style-type: none"> • No active control 	<p>Present</p> <ul style="list-style-type: none"> • Ex-post settlement based on sharing coefficients determined by the community => static heuristic for now but would also like to implement dynamic coefficients.
			<p>Future</p> <ul style="list-style-type: none"> • Would like to implement direct P2P with individual pricing per bilateral exchange => P2P model • Provide external flexibility services => market clearing algorithm to select flexible volumes amongst community members.

The different use cases outlined in Table 2 are linked back to the algorithms identified in Section 3.2, to show that the proposed algorithms cover the full range of use cases wished by the demo sites.

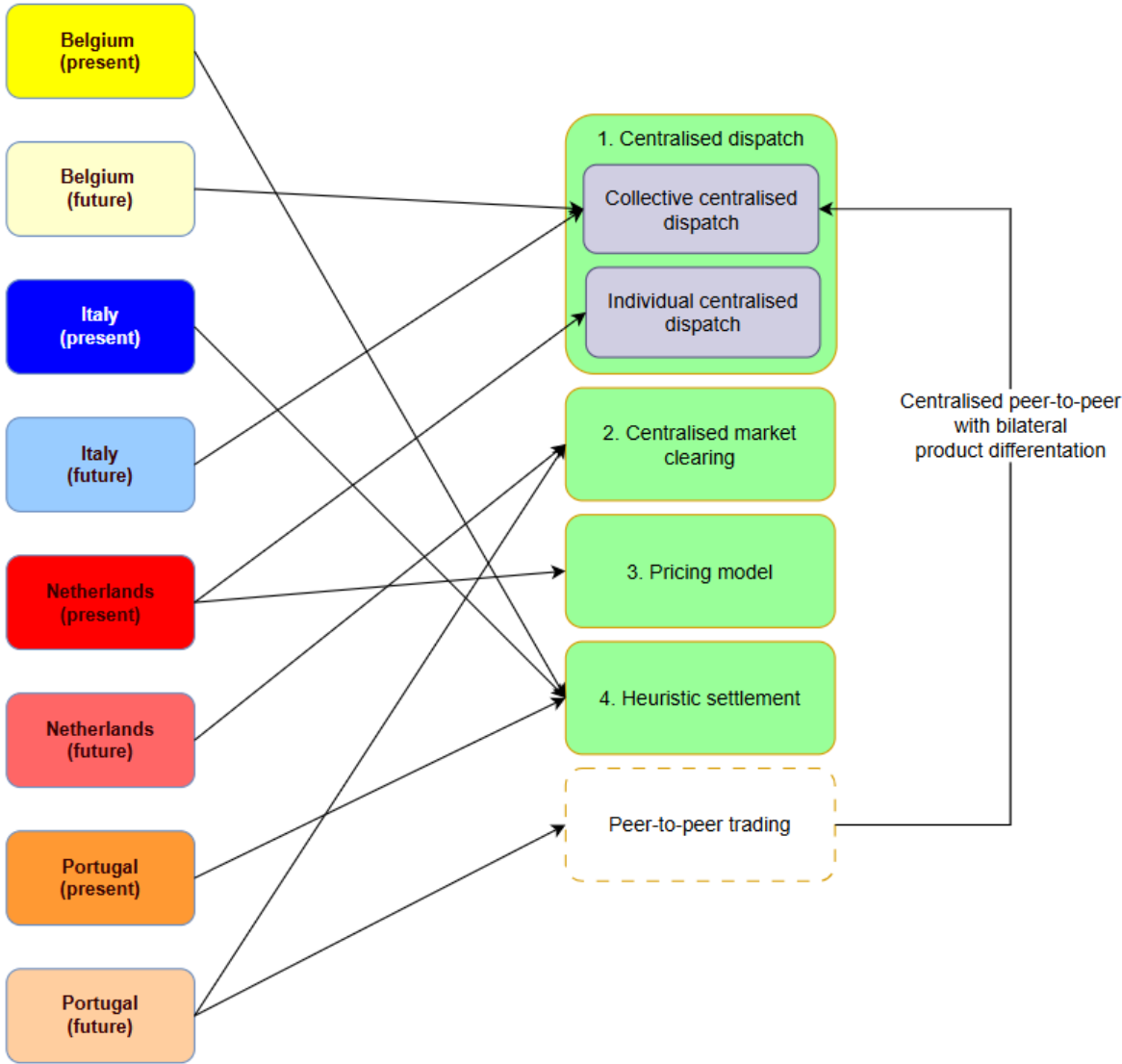


Figure 16: Link between demo site use cases and developed algorithms in WP4

Figure 16 shows that the present use cases considered by the demo sites are mainly covered by the algorithms 2 or 3. In any of these use cases, there is no energy sharing ahead of dispatch time, and members either reallocate benefits of being part of the community ex-post during the settlement phase (BE, IT, PT) or optimise their individual schedule based on signals sent by the energy CM (NL).

In the future, the settlement phase could be implemented as a centralised optimisation problem, in which case the centralised dispatch problem would be relevant. Algorithms 1 and 2 are mainly important to implement for future use cases, once energy sharing is considered.

For the Portuguese case, where P2P exchanges are considered in the future, the proposed algorithms do not address the possibility of doing so in a decentralised way. However, the centralised dispatch algorithm can be adapted to include bilateral peer preferences, in which a centralised P2P approach can be modelled by the proposed algorithms.

Overall, we can conclude that the proposed set of algorithms addresses most needs of the demo sites, without containing a too large number of algorithms to implement.

4 Specifications for algorithm implementations

4.1 Common notation

Irrespective of the algorithms considered, some common elements can be defined across all algorithms. These elements are introduced in this first section.

4.1.1 Time parameters

The time parameters used are shown in Table 3 and illustrated in Figure 17.

Table 3: Parameters used for common notation across algorithms

Parameter	Equivalent DTO attribute	Description
T		Discrete set of time indices
t		Time index
Δt	timestep_minutes	Dispatch/market time (default at 15 minutes), adjustable between 15 minutes to an hour.
h	horizon_hours	Optimisation horizon (default at 24 hours)
m		Interval between two dispatch/market resolutions (default at 24 hours)
d		Clearing time ahead of market dispatch

Optimisation horizons below 24 hours allow to have intraday adjustments based on updated forecast information, while optimisation horizons above 24 hours avoid short-sighted strategies. This is particularly true for storage assets, which can improve their operational profile from long-term planning, even though only the first part of the optimisation horizon is kept in the operational phase.

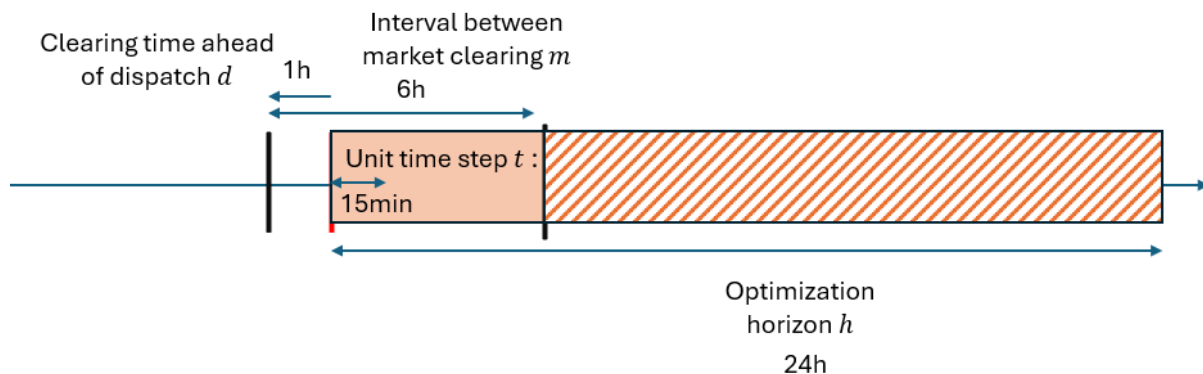


Figure 17: Time parameters used in algorithms

This implies that only the timesteps between two optimisation calls are dispatched as planned in the optimisation program. Subsequent timesteps will be modified by the optimisation algorithm based on updated information on the system.

4.1.2 General assumptions and hypotheses

General assumptions

Several assumptions were made regarding the context and environment of the EC, namely:

- An EC is composed of a CM and members (also referred to as peers).
- The CM is responsible for coordinating energy and benefit allocation within the community, acting as the intermediary between all members. Specific responsibilities may vary depending on the algorithm implemented.
- A member may belong to a household, and a household can include multiple members. Each community member can possess different types of energy assets, such as non-flexible loads, PV panels, electric vehicles (EVs), flexible loads, and heat pumps.
- A meter measures the total energy production and consumption of the assets connected to it (e.g., production units, storage systems, or flexible/non-flexible loads). Consequently, a community member may have multiple meters for different assets — for example, one for household consumption and another for EV charging. Each meter is associated with a contract and a distribution point. In the case of collective assets, a single meter may be linked to multiple contracts, and the share of each contract in the asset must be defined.
- A contract belongs to a specific contract category, determined by the pricing structure (e.g., fixed-rate blocks, peak/off-peak hours, or variable pricing). It is linked to an energy supplier and limited by a maximum power threshold, which may represent either

subscription limits or physical transmission capacity. Each supplier defines its own energy supply and resale prices, as well as capacity tariffs. Thus, a meter serves as the physical measurement device associated with an economic object (the contract) used by the supplier for billing purposes.

- The EC is exposed to external market conditions: the energy offtake and injection prices from the grid are known for each contract and each time step prior to the dispatch resolution.

Modelling assumptions

Modelling assumptions are made throughout this work:

- It is assumed that predictions are accurate and that actual outcomes do not significantly deviate from the forecasts.
- The developed algorithms do not consider network effects and voltage constraints.
- Households in the community are exempt from transmission network charges and taxes for energy shared within the community. However, they must pay for the distribution network costs for utilising the distribution network.
- The community power exchanges with the grid are limited to promote self-consumption and reduce power losses.
- Simplifying assumptions are taken to represent assets in this work. In order to be reusable, these asset models cannot include too specific parameters, which would overfit the model to a specific problem at hand. On the other hand, the simplifications taken should not prevent to developed more complex asset models for case-specific implementations.

To handle this, an object-oriented approach is taken, where assets inherit from parent classes which are more generic. More advanced representations of assets are made possible through this work by creating child classes which complement and potentially overwrite features from the parent class.

Asset categories

Due to the object-oriented approach, an inheritance tree can be built to group assets owned by community members (or collectively owned by the community) into categories. First, they are classified as either non-flexible assets or flexible assets. Non-flexible assets include resources whose operation cannot be controlled, such as non-flexible load assets and variable production assets like photovoltaic (PV) systems. Flexible assets, on the other hand, represent controllable resources and include flexible load assets such as Heating, Ventilation, and Air Conditioning (HVAC) systems, dispatchable production assets, and storage devices like battery storage (BS) and EVs (Figure 18).

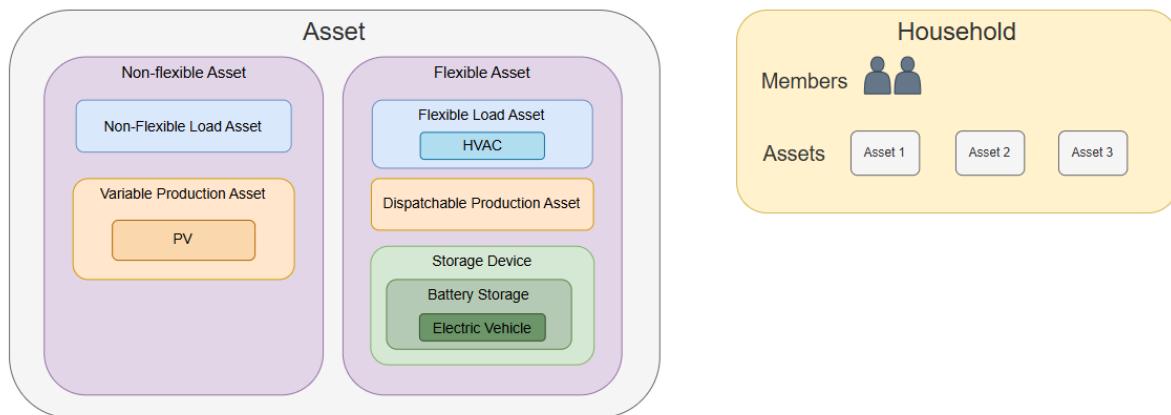


Figure 18: Asset classification and household class representation

4.1.3 Common library for all algorithms

The concepts previously introduced are transcribed into python classes, whose parameters (or attributes) are presented in Table 4 to Table 9.

Table 4: Contract class parameters

<i>Parameter</i>	<i>Equivalent DTO attribute</i>	<i>Description</i>
	category	Contract category
	id	Contract ID
	supplier	Supplier associated with the contract
\overline{P}_c^{inj}	max_power_injected_kw	Maximum injected power allowed for contract c
\overline{P}_c^{with}	max_power_consumed_kw	Maximum withdrawn power allowed for contract c

Table 5: Meter class parameters

Parameter	Equivalent DTO attribute	Description
	id	Meter ID
	linked_distribution_point	Distribution point linked to the meter
	contracts	List of contracts associated with the meter
α_i	contract_shares	List of shares (from each contract in shared assets)
	power_kw	Physically exchanged electricity

Table 6: Supplier class parameters

Parameter	Equivalent DTO attribute	Description
	supply_prices	
	resale_prices	
	capacity_tariffs	

Table 7: Community member class parameters

Parameter	Equivalent DTO attribute	Description
	id	Member ID
	meters	List meters associated to the community member

Table 8: Household class parameters

Parameters	Equivalent DTO attribute	Description
	community_members	List of community members belonging to the household
	assets	Associated list of Assets
	risk_aggressiveness	Risk aggressiveness of the household

A community represented by the CM (Energy Community Manager):

Table 9: Energy Community Manager parameters

Parameters	Equivalent DTO attribute	Description
	id	Community ID
	contract	Contract associated to the community

4.2 Algorithm 1: Centralised energy management system

The first algorithm category handles centralised energy management systems. When executed by the CM, it takes as inputs some individual parameters of the members and solves the problem at the community scale, in a centralised way. This centralised energy management system can be implemented in different ways, depending on the time of optimisation and the involvement of the community members in the energy sharing process, as seen in the models presented in [1]. These different approaches are described below.

Algorithm 1.1 models a centralised energy management system at the community level (as shown in Figure 19), where only the centralising agent (CM) can optimise. In the centralised version of the Capacity, Energy and Allocation model of [1], households share their operational constraints and desired setpoints with the CM. The CM has control over all collective assets and therefore has access to their parameters and needs to model them.

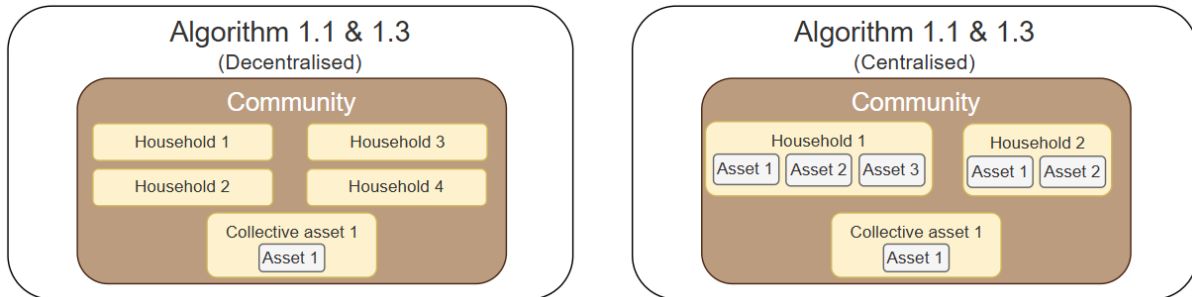


Figure 19: Block representation of Algorithm 1.1 (Decentralised vs Centralised versions)

Algorithm 1.2 (Figure 20) is used in a situation where community members are trading among each other with set preferences. Households are modelled in high-level, without need to go down to the assets.

Algorithm 1.3 is executed to achieve a fair and efficient collective benefit allocation ex-post to the operational time. It relies on an iterative approach, where the centralised energy management algorithm 1.1 is repeated over different subsets / coalitions within the community, to identify the marginal contribution of each community member [1]. The algorithm therefore mainly differentiates itself from 1.1 through the need to implement an architecture which allows an iterative approach. Finally, Algorithm 1.4 (Figure 20) models a centralised energy management system at the household level. Only one household is represented with all its assets.

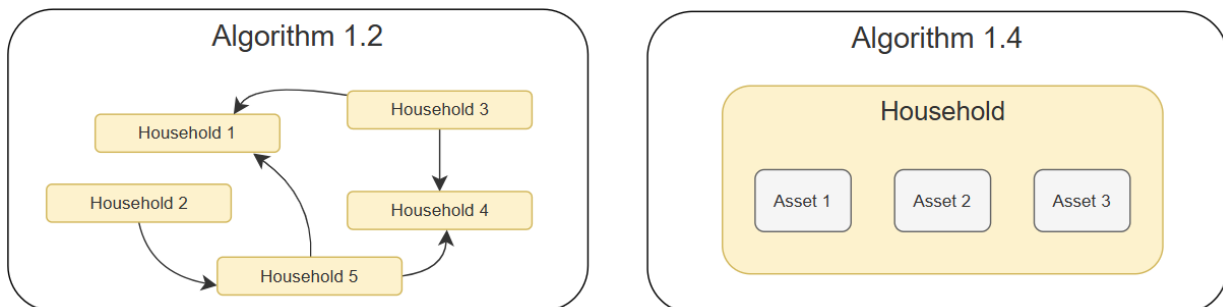


Figure 20: Block representation of Algorithm 1.2 and 1.4

All the above algorithms have some common elements that can be formalised. Hence, a common library to all dispatch algorithms is defined.

4.2.1 Common library for centralised energy management system algorithms

The common library to solve the centralised energy management system problems is introduced in this section. Unlike following sections, which only focus on the interface structure of the algorithms, this section also describes how generic assets reused in different algorithms are modelled. This is because these asset models are considered algorithm-agnostic and will be reused in different contexts. The classes for these asset models therefore contain the OFs and the constraints which are specific to these assets, and which will have to be included in whichever algorithm these assets are implemented in. These objectives and constraints are therefore explicitly written in the rest of this section.

The asset models remain however quite generic and might not be applicable to all cases. Rather than being exhaustive, the proposed simplified models try to accommodate for further changes if needed. These potential changes can simply complement or overwrite the models presented here.

4.2.1.1. Sets and indices

Table 10 presents the sets and indices used in the common library.

Table 10: Notation for sets and indices in the common library

Notation	Description
Ω^I	Set of households
Ω^A	Set of assets
Ω_i^a	Set of households linked to asset a
Ω_a^c	Set of assets linked to contract c
n	Indices related to member n
i	Indices related to household i
c	Indices related to contract c
ec	Indices related to the EC

For simplicity of representation, a collective asset is modelled in the same way as other assets, meaning it is treated as the sole asset of its corresponding household.

4.2.1.2. Input data

The input data common to all algorithms are described below.

Input time series

The following time series serve as input of the developed algorithms. It is worth mentioning that, for each algorithm, not all the inputs are needed, and can therefore be left empty in such case.

- Load
 - Forecast inflexible load profiles $D_{t,i}$ for each household i at each time step t , in [kWh].
 - Forecast flexible load profiles $D_{t,i}^{fl}$, in [kWh].
- Solar PV generation
 - Solar forecast profiles $f_{t,i}^{PV}$ for each site at each time step, in [kWh]. The solar forecast is specific to each household as some panels can have different directions.
- Flexible Availability
 - Binary values $A_{t,i}$: $A_{t,i} = 1$ indicates if a flexible asset (EV, battery, etc.) is available for charging. $A_{t,i} = 0$ otherwise.
- Outside temperature
 - Outside temperature profile T_t^{out} is given for the whole community, in [°C].
- Retail price
 - Import retail price $\lambda_{t,i}^{imp}$ defined by the energy supplier of each household i , in [€/kWh]
 - Export retail price $\lambda_{t,i}^{exp}$ defined by the energy supplier of each household i , in [€/kWh]
- Capacity tariffs
 - Distribution network cost based on consumers' behaviour or contract.

Remark: for consistency, when a household or the community are exchanging with the grid, powers and prices will be denoted export and import. On the other hand, when dealing with exchanges within the community, those same quantities will be referred to buying and selling.

Input parameters

The input parameters shown below to model the assets correspond to simplified asset representations. Some of these parameters may be overwritten if more complex site-specific asset models need to be developed. These more complex models will be based on asset classes which inherit the parameter inputs from these more simplified models, but can overwrite them if needed.

The parameters used to model batteries are presented in Table 11.

Table 11: Parameters used for battery modelling

Parameters	Equivalent DTO attribute	Description
	id	BS ID
	meter	Associated meter
\underline{p}_i^{BS}	p_min_w	BS minimum power
\overline{P}_i^{BS}	p_max_w	BS rated power capacity
$\eta_{daily\ cycle}$	max_cycles	Maximum number of cycles per day
η_i^{BS}	charge_efficiency	Charging efficiency (discharging efficiency assumed equal)
δ_i^{BS}	self_discharge_rate	BS self-discharge rate
\overline{E}_i^{BS}	storage_capacity	BS energy capacity
\underline{SoC}_i^{BS}	soc_min	Minimum SoC
\overline{SoC}_i^{BS}	soc_max	Maximum SoC
$A_{t,i}^{BS}$	availability_flex	Flexible availability
α_{cyc}^{BS}	weight_cycling	OF penalization
SoC_i^{BS*}	soc_target	Minimum target SoC
$t^{BS,*}$	t_target	Time by which the minimum target SoC must be achieved

SoC_{init,t_0}^{BS}	soc_init	Initial SoC for initial time t_0
-----------------------	----------	------------------------------------

The parameters used to model the EV parameters, or rather, the parameters for the EV battery, are shown in Table 12.

Table 12: Parameters used for EV modelling

Parameters	Equivalent DTO attribute	Description
	id	EV ID
	meter	Associated meter
\underline{P}_i^{EV}	p_min_w	EV minimum power
\overline{P}_i^{EV}	p_max_w	EV rated power capacity
$\eta_{\text{daily cycle}}$	max_cycles	Maximum number of cycles per day
η_i^{EV}	charge_efficiency	Charging efficiency (discharging efficiency assumed equal)
δ_i^{EV}	self_discharge_rate	EV self-discharge rate
\overline{E}_i^{EV}	storage_capacity	EV energy capacity
\underline{SoC}_i^{EV}	soc_min	Minimum State of Charge (SoC)
\overline{SoC}_i^{EV}	soc_max	Maximum SoC
$A_{t,i}^{EV}$	availability_flex	Flexible availability
α_{cyc}^{EV}	weight_cycling	OF penalization
SoC_{init,t_0}^{EV}	soc_init	Initial SoC for initial time t_0
ξ_{EV}	weight_comfort	Penalization weight for not meeting the mobility-related SoC target
SoC_i^{EV*}	soc_target	Target state-of-charge

$t^{EV,*}$	t_target	Desired time by which the target SoC must be achieved
	charger	Associated EV charger

The parameters used for the EV charger are shown in Table 13.

Table 13: Parameters used for EV charger modelling

<i>Parameters</i>	<i>Equivalent DTO attribute</i>	<i>Description</i>
	id	EV charger ID
	meter	Associated meter
\bar{C}_i^{EV}	c_max	EV charger maximum power capacity

The parameters used for PV modelling are shown in Table 14.

Table 14: Parameters used for PV modelling

<i>Parameters</i>	<i>Equivalent DTO attribute</i>	<i>Description</i>
	id	ID
	meter	Associated meter
$f_{t,i}^{PV}$	forecasted_profile	Solar forecast
\bar{P}_i^{PV}	rated_capacity	PV rated capacity [kW]
ξ_{PV}	weight_curtailment	Penalization of solar curtailment

The parameters used for heat pump (HVAC) modelling are shown in Table 15.

Table 15: Heat pump parameters

<i>Parameters</i>	<i>Equivalent DTO attribute</i>	<i>Description</i>
	id	ID
	meter	Associated meter
\overline{P}_i^{HVAC}	p_max_w	Maximum HVAC power
\underline{P}_i^{HVAC}	p_min_w	Minimum HVAC power
	baseline_forecast	Operational schedule
T_i^{in*}	setpoint_temp	Setpoint temperature
\overline{T}_i^{max}	max_temp	Maximum temperature
\underline{T}_i^{min}	min_temp	Minimum temperature
T_{init}^{in}	indoor_temp	Initial indoor temperature
θ_i	thermal_inertia	Thermal inertia of building
η_i^{HVAC}	efficiency_factor	Efficiency factor
ξ_{ac}	weight_comfort	Penalization weight for not meeting the desired temperature
T_t^{out}	outdoor_temp	Outdoor temperature

The parameters used for flexible load modelling are shown in Table 16.

Table 16: Parameters used for flexible load modelling

<i>Parameters</i>	<i>Equivalent DTO attribute</i>	<i>Description</i>
	id	ID
	meter	Associated meter
ξ_f		Penalization of load shifting

E_i^{fl}	total_expected_energy_consumption	Total expected energy consumption on specific time period
$A_{t,i}$	availability_flex	Flexible availability, containing t_{min} and t_{max} values associated to E_i^{fl}
$D_{t,i}^{fl}$	baseline_forecast	Expected operational schedule

The parameters used for unflexible load modelling are shown in Table 17.

Table 17: Parameters used for unflexible load modelling

<i>Parameters</i>	<i>Equivalent DTO attribute</i>	<i>Description</i>
	id	ID
	meter	Associated meter
$D_{t,i}$	forecasted_profile	Forecast profile of the inflexible load

The Value of Lost Load (VoLL) is used to ensure feasibility of the optimisation problem, by allowing load shedding Table 18.

Table 18: Parameters used to ensure feasibility of the optimisation problem

<i>Parameters</i>	<i>Description</i>
λ^{VoLL}	Value of lost load

4.2.1.3. Variables

The following variables are introduced:

- Battery Storage:

$SoC_{t,i}^{BS}$: SoC of household i at time t

$P_{t,i}^{BS,ch}$: Power charged in the battery

$P_{t,i}^{BS,dis}$: Power discharged in the battery

$z_{t,i}$: Binary variable associated to battery (to prevent simultaneous charging and discharging)

- EV:

$SoC_{t,i}^{EV}$: SoC of EV battery

$P_{t,i}^{EV,ch}$: Power charged in the EV battery

$P_{t,i}^{EV,dis}$: Power discharged in the EV battery

$y_{t,i}$: Binary variable associated to EV battery (to prevent simultaneous charging and discharging)

- PV:

$P_{t,i}^{PV}$: PV power production

- HVAC:

$T_{t,i}^{in}$: Indoor temperature

$P_{t,i}^{HVAC}$: Heat pump power

- Loads:

$d_{t,i}^{nf}$: Inflexible (non-flexible) load

$d_{t,i}^f$: Flexible load

- Grid exchanges at the household level

$P_{t,i}^{imp}$: Power imported from the grid

$P_{t,i}^{exp}$: Power exported to the grid

$P_{t,c}^{imb}$: Imbalance power associated to contract c

$P_{t,c}^+$: Positive part of imbalance power associated to contract c

$P_{t,c}^-$: Negative part of imbalance power associated to contract c

- Grid exchanges and prices at the community level

$P_t^{imp,ec}$: Power imported to the community from the grid

$P_t^{exp,ec}$: Power exported from the community to the grid

$\lambda_{t,i}^{buy}$: Internal buying price of the community (depending on the algorithm, it could be an input parameter)

$\lambda_{t,i}^{sell}$: Internal selling price of the community (depending on the algorithm, it could be an input parameter)

4.2.1.4. Objective functions

The Objective Functions (OF) specific to each asset type are introduced:

- Minimise battery cycle cost: during BS operation, the prosumer aims to minimise the cycling cost to avoid excessive charge and discharge cycles, which can degrade the battery's lifetime.

$$f_i^{BS} = \alpha_{cyc} \sum_{t \in T} \frac{(\eta_i^{BS} P_{t,i}^{BS,ch} + \frac{1}{\eta_i^{BS}} P_{t,i}^{BS,dis}) \Delta t}{2\bar{E}_i^{BS}}$$

- Minimise EV cycle cost and the deviation from the targeted charging level:

$$f_i^{EV} = \alpha_{cyc} \sum_{t \in T} \frac{(\eta_i^{EV} P_{t,i}^{EV,ch} + \frac{1}{\eta_i^{EV}} P_{t,i}^{EV,dis}) \Delta t}{2\bar{E}_i^{EV}} + \xi_{EV} (SoC_{tEV,*}^{EV} - SoC_i^{EV*})^2$$

- Minimise the thermal discomfort of the household:

$$f_i^{HP} = \xi_{ac} \sum_{t \in T} (T_{i,t}^{in} - T_i^{in*})^2$$

- Minimise load shedding:

$$f_i^{nf} = \lambda^{VoLL} \sum_{t \in T} (D_{t,i} - d_{t,i}^{nf})^2$$

- Minimise flexible load deviation:

$$f_i^f = \xi_f \sum_{t \in T} (D_{t,i}^f - d_{t,i}^f)^2$$

- Minimise curtailment:

$$f_i^{PV} = \xi_{PV} \sum_{t \in T} (f_{t,i}^{PV} - P_{t,i}^{PV})^2$$

4.2.1.5. Constraints

Balance constraints

- At the household level:

$$P_{t,i}^{PV} + P_{t,i}^{BS,dis} + P_{t,i}^{EV,dis} + P_{t,i}^{HVAC} - P_{t,i}^{BS,ch} - P_{t,i}^{EV,ch} - d_{t,i}^{nf} - d_{t,i}^f = P_{t,i}^{exp} - P_{t,i}^{imp}$$

- At the community level (power not compensated between members is imported / exported outside of the community):

$$\sum_{i \in \Omega^I} (P_{t,i}^{PV} + P_{t,i}^{BS,dis} + P_{t,i}^{EV,dis} + P_{t,i}^{HVAC} - P_{t,i}^{BS,ch} - P_{t,i}^{EV,ch} - d_{t,i}^{nf} - d_{t,i}^f) = P_t^{exp,ec} - P_t^{imp,ec}$$

Technical constraints

BS

- SoC dynamics:

$$SoC_{t+1,i}^{BS} = (1 - \delta_i^{BS}) SoC_{t,i}^{BS} + \frac{\Delta t}{E_i^{BS}} (\eta_i^{BS} P_{t,i}^{BS,ch} - \frac{1}{\eta_i^{BS}} P_{t,i}^{BS,dis}) \quad \forall t \in T$$

$$SoC_{t_0+1,i}^{BS} = (1 - \delta_i^{BS}) SoC_{init,t_0}^{BS} + \frac{\Delta t}{E_i^{BS}} \left(\eta_i^{BS} P_{t_0,i}^{BS,ch} - \frac{1}{\eta_i^{BS}} P_{t_0,i}^{BS,dis} \right) \quad t = t_0$$

- Charging and discharging constraints where the binary variable prevents simultaneous charging and discharging:

$$z_{t,i} \underline{P}_i^{BS} \leq P_{t,i}^{BS,ch} \leq z_{t,i} \overline{P}_i^{BS} \quad \forall t \in T$$

$$(1 - z_{t,i}) \underline{P}_i^{BS} \leq P_{t,i}^{BS,dis} \leq (1 - z_{t,i}) \overline{P}_i^{BS} \quad \forall t \in T$$

- BS state-of-charge must be lower than the maximum SoC allowed and higher than the minimum:

$$\underline{SoC}_i^{BS} \leq SoC_{t,i}^{BS} \leq \overline{SoC}_i^{BS} \quad \forall t \in T$$

- Minimum SoC reached at target time

$$SoC_{i,t_{BS,*}}^{BS} \geq SoC_i^{BS*}$$

- Daily cycle limitation

$$\sum_{t \in T} \frac{(\eta_i^{BS} P_{t,i}^{BS,ch} + \frac{1}{\eta_i^{BS}} P_{t,i}^{BS,dis}) \Delta t}{2 \overline{E}_i^{BS}} \leq \eta_{daily\ cycle} \quad \forall t \in T$$

EV:

- SoC dynamics:

$$SoC_{t+1,i}^{EV} = (1 - \delta_i^{EV}) SoC_{t,i}^{EV} + \frac{\Delta t}{\overline{E}_i^{EV}} (\eta_i^{EV} P_{t,i}^{EV,ch} - \frac{1}{\eta_i^{EV}} P_{t,i}^{EV,dis}) \quad \forall t \in T$$

$$SoC_{t_0+1,i}^{EV} = (1 - \delta_i^{EV}) SoC_{init,t_0}^{EV} + \frac{\Delta t}{\overline{E}_i^{EV}} \left(\eta_i^{EV} P_{t_0,i}^{EV,ch} - \frac{1}{\eta_i^{EV}} P_{t_0,i}^{EV,dis} \right) \quad t = t_0$$

- Charging and discharging limits where the binary variable prevents simultaneous charging and discharging. The charging power must be lower than the minimum of the maximum EV power and the EV charger capacity. It must also be higher than the minimum EV power.

$$y_{t,i} \underline{P}_i^{EV} \leq P_{t,i}^{EV,ch} \leq y_{t,i} \min(\overline{P}_i^{EV}, \overline{C}_i^{EV}) \quad \forall t \in T$$

$$(1 - y_{t,i}) \underline{P}_i^{EV} \leq P_{t,i}^{EV,dis} \leq (1 - y_{t,i}) \min(\overline{P}_i^{EV}, \overline{C}_i^{EV}) \quad \forall t \in T$$

- SoC limits

$$\underline{SoC}_i^{EV} \leq SoC_{t,i}^{EV} \leq \overline{SoC}_i^{EV} \quad \forall t \in T$$

Heat pump:

- Inside temperature computation (split into two equations, depending on cooling or heating)

- Cooling ($T_t^{out} > T_{t,i}^{in}$)

$$T_{t,i}^{in} = \theta_i T_{t-1,i}^{in} + (1 - \theta_i)(T_{t-1}^{out} - \eta_i^{HVAC} P_{t,i}^{HVAC}) \quad \forall t \in T$$

- Heating ($T_t^{out} \leq T_{t,i}^{in}$)

$$T_{t,i}^{in} = \theta_i T_{t-1,i}^{in} + (1 - \theta_i)(T_{t-1}^{out} + \eta_i^{HVAC} P_{t,i}^{HVAC}) \quad \forall t \in T$$

- Inside temperature limits

$$\underline{T}_i^{min} \leq T_{t,i}^{in} \leq \overline{T}_i^{max} \quad \forall t \in T$$

- HVAC power limits

$$\underline{P}_i^{HVAC} \leq P_{t,i}^{HVAC} \leq \overline{P}_i^{HVAC} \quad \forall t \in T$$

PV:

- PV production is restrained by the associated forecast:

$$0 \leq P_{t,i}^{PV} \leq f_{t,i}^{PV} \quad \forall t \in T$$

Inflexible load:

- Inflexible load is limited by the consumption profile:

$$0 \leq d_{t,i}^{nf} \leq D_{t,i} \quad \forall t \in T$$

Flexible demand:

- Flexible load is limited by the consumption forecast profile:

$$0 \leq d_{t,i}^f \leq D_{t,i}^{fl} \quad \forall t \in T$$

- Accumulated flexible load power over the desired time period should be higher than a defined threshold:

$$\sum_{t_{min} \leq t \leq t_{max}} d_{t,i}^f \Delta t \geq E_i^{fl}$$

Grid constraints

Contract limit exchanges with the grid

- For **each contract linked to the household**, the imbalance is defined as the difference of the aggregated production power and the aggregated demand power of all assets linked to this contract:

$$\sum_{\substack{i \in \Omega_a^c \cap \Omega_i^a \\ prod}} P_{t,i} - \sum_{\substack{i \in \Omega_a^c \cap \Omega_i^a \\ load}} d_{t,i} = P_{t,c}^{imb} \quad \forall c \in \Omega_c^i$$

- The imbalance power is separated in positive and negative parts:

$$P_{t,c}^{imb} = P_{t,c}^+ - P_{t,c}^- \quad \forall t \in T$$

- Imbalance power is limited by the contract limits:

$$P_{t,c}^+ \leq \bar{P}_c^{exp} \quad \forall t \in T$$

$$P_{t,c}^- \leq \bar{P}_c^{imp} \quad \forall t \in T$$

- The imbalance at the household level $P_{t,i}^{imb}$ is the sum of the imbalance of each contract.

$$\sum_{c \in \Omega_c^i} P_{t,c}^{imb} = P_{t,i}^{imb} \quad \forall t \in T$$

- The total imbalance is decomposed in import and export power (linked to the balance constraint).

$$P_{t,i}^{imb} = P_{t,i}^{exp} - P_{t,i}^{imp} \quad \forall t \in T$$

Community constraints

Grid constraint at the community scale (with $\bar{P}^{imp,ec}$ and $\bar{P}^{exp,ec}$ the contract limits associated to the community):

$$0 \leq P_t^{imp,ec} \leq \bar{P}^{imp,ec} \quad \forall t \in T$$

$$0 \leq P_t^{exp,ec} \leq \bar{P}^{exp,ec} \quad \forall t \in T$$

4.2.2 Algorithm 1-1: Centralised energy management system for Community Energy Management System

This energy sharing algorithm is being developed in a context with an energy CM having the possibility to centrally control or at least schedule the assets within the community, taking into account the constraints and objectives of the community members.

4.2.2.1. General formulation and hypotheses

It is assumed that the community has access to the energy offtake and injection prices of each household for each time step prior to dispatch resolution.

It is assumed that the CM can either operate all energy resources for the purpose of achieving the member level preferences (Model 2.2) or can operate only the collective assets (Model 2.1). In the latter, members are only sharing fixed injection and offtake profiles.

4.2.2.2. Objective

The algorithm is executed to determine the optimal energy injections and withdrawals of available storage and flexible assets within the community, with the objective of minimising the total energy cost over the dispatch horizon. Assets are scheduled so as to minimise the net imports of the community with regard to the grid.

Additional objectives may be incorporated to address specific community priorities or constraints. Context-specific examples of such extensions are provided in Appendix A.

4.2.2.3. Input data

All assets' parameters and time series are described in Section 4.2.1.2 are required for this algorithm.

4.2.2.4. Constraints

The problem is restricted by a set of constraints (introduced in Section 4.2.1.5).

In Model 2.1, members only share their injection/ offtake profiles. Hence, the following constraints are needed:

- Technical constraints of shared assets within the community;
- balance constraint at the *community level*;
- community grid constraints.

In Model 2.2, members give control of their DER to the CM. Additional constraints are needed:

- Technical constraints of all individual and shared assets within the community;
- balance constraint at the *community level*;
- individual grid constraints of each household;
- community grid constraints.

4.2.2.5. Decision variables/ Output

Outputs for Model 2.1 are:

- the power injections and withdrawals of each household,
- the power injections and withdrawals for each collectively owned asset,
- the total power exchanged with the grid.

Outputs for Model 2.2 are:

- the power injections and withdrawals for each asset (individual and collectively owned),
- the total power exchanged with the grid.

4.2.3 Algorithm 1-2: Centralised energy management system with “Peer-to-peer” preferences

Algorithm 1.2 solves a centralised energy management system problem, but with peers defining preferences to exchanges with other peers. This weighting of exchange preferences leads to bilateral prices between community members, replicating P2P sharing patterns, although cleared in a centralised way.

4.2.3.1. General assumptions

The following assumptions are made to model a centralised dispatch with P2P preferences:

- Peers are assumed to have full control over their consumption and DER assets.
- Each peer is equipped with a home energy management system (HEMS) which helps optimise its energy usage.
- Each peer in any given time period can either act as a producer or as a consumer in bilateral trade.
- Peers that are consumers at a given time period can buy energy from their energy supplier if their demand cannot be met by P2P producers.
- Peers that are producers at a given time period can sell energy to their energy supplier if their production cannot be consumed within the P2P market.

4.2.3.2. Specific nomenclature

Sets & indices

Table 19 shows the sets and indices used for the centralised energy management system with P2P preferences.

Table 19: Sets and indices used for Centralised energy management with P2P preferences

<i>Notation</i>	<i>Description</i>
Ω	Set of members
Ω_p	Set of producers
Ω_c	Set of consumers
ω_n	Set of trading partners of peer n
Ω_{pRES}	Subset of producers with renewable generation Ω_p

G	Set of criteria
n	indices referring to a peer
g	indices referring to a criterion

Parameters

Each peer has a quadratic cost function, which can be generated using the attributes in Table 20.

Table 20: Attributes used for modelling the quadratic cost function of peers

<i>Notation</i>	<i>Equivalent DTO attribute</i>	<i>Description</i>
a_n	a	Quadratic coefficient of cost function
b_n	b	Linear coefficient of cost function
d_n	c	Constant coefficient of cost function

Each peer can express a preference for other peers through a product differentiation parcel, which is expressed based on the attributes shown in Table 21.

Table 21: Attributes used for expressing the product differentiation parcel in the OF

<i>Notation</i>	<i>Equivalent DTO attribute</i>	<i>Description</i>
ω_n	trading_partners	List of trading partners of peer n
c_n^g	trade_coefficient	Cost based on criterion g
$\gamma_{n,m}^g$	trade_characteristic	Preference of peer m to trade with peer n based on criterion g

Finally, combining the information from the tables above, the attributes used to represent each peer in the optimisation problem are shown in Table 22.

Table 22: Attributes for modelling peers in the energy management system problem with P2P preferences

<i>Notation</i>	<i>Equivalent DTO attribute</i>	<i>Description</i>
	id	Peer ID
	assets	List of assets of peer n
C_n	cost_function	Quadratic cost function associated to peer n
\overline{C}_n	product_differentiation_parcel	Product differentiation parcel associated to peer n
\underline{P}_n	min_availability	Lower flexibility bound
\overline{P}_n	max_availability	Upper flexibility bound
	direction	Direction of flexibility

4.2.3.3. Decision variables

Decision variables of this problem are defined below:

- P_n : Active power of peer n
- $P_{n,m}$: Bilateral trade between peer n and peer m
- $\lambda_{n,m}$: Dual variable representing the price of trade between n and m

The associated output is therefore the power exchanged between each peer and the price associated with that trade.

4.2.3.4. Objective

The P2P market algorithm minimises the procurement cost to meet the net demand for electricity. Unlike previous algorithms, the stated purchase price for each community member is expressed as a function of the member from which the electricity is purchased from, effectively providing bilateral P2P preferences [1]. This preference is stated as a “product

differentiation” cost, in addition to the member’s own cost function, which is expressed as a quadratic function of power [1].

The optimisation problem then calculates the optimal schedule at a community level based on the bilateral prices stated by members.

4.2.3.5. Input parameters

General parameters

- The community is identified with an identifier (ID), as well as each peer.

Peer information

- Each peer shares the cost coefficients of its cost function and the trade coefficient associated with each criterion to the CM. Trade characteristics with all other peers within the community should also be defined and shared.
- The active power P_n can be flexible for any producer or consumer. In that case, both the upper \bar{P}_n and lower \underline{P}_n bounds of P_n need to be defined.

4.2.3.6. Constraints

The physical constraints of the assets are included as in the previous algorithms. For each of these constraints, it is the sum of bilateral trades which must be taken into account to evaluate the physical position of each asset.

In addition, constraints must be included to make sure that the volume sold corresponds to the volume bought for each bilateral trade. The dual variable of these constraints provides the price of trade between community members, referred to here as peers.

4.2.4 Algorithm 1-3: Collective benefit allocation (settlement)

Once the community has physically consumed/produced its electricity, with or without optimal scheduling, energy sharing can be applied ex-post in the settlement phase, by netting energy bills amongst producers and consumers within the community. This energy netting yields a collective benefit, which must be distributed across the community members.

4.2.4.1. General formulation and hypotheses

It is considered realistic that billing may be reprocessed by the community after the supplier's initial billing, allowing payback to be requested from suppliers for certain contracts based on an optimized community-level billing process. Therefore, the benefit allocation is performed

ex-post, considering the realization/dispatch of all devices within the community. Note that it will also consider the outcome of a flexibility mechanism if it exists.

Allocation mechanisms considered are those that are efficient, i.e., the total cost/benefit of the community is entirely distributed to the members, based on cooperative game-theoretic theory, explained below.

4.2.4.2. Allocation mechanisms

Two cost allocation mechanisms, both built upon Algorithm 1.1 and based on the work in [1], are considered:

- **Worst-case excess minimisation** [6]: The worst-case excess minimisation is a fair value-sharing method that allocates the total benefit of cooperation so that the most dissatisfied subgroup (coalition) is as satisfied as possible. In other words, it finds an allocation that minimises the largest potential gain any coalition could achieve by leaving the community.
- **Shapley value** [7]: The Shapley value is a cooperative game theory method used to fairly allocate collective benefits or costs among participants based on their individual contributions. Applied to an EC, it quantifies each member's contribution to the community's overall cost savings or benefits. By evaluating all possible combinations of participants, the Shapley value ensures that each member receives a fair share of the total benefit proportional to their marginal impact on the community's performance.

In both cases, iterations of the collective energy management problem have to be made over different coalitions (subsets of the community), in order to determine the redistribution coefficients based on the outcomes of these different possible coalitions.

4.2.4.3. Input parameters

The following inputs are needed:

- The total benefit (or cost) of the community. It can also include the remuneration of flexibility provision;
- Offtake and/or injection profiles of each community member;
- Centralised optimisation problem definition (objectives and constraints) and related parameters as defined in Section 4.2.2 for Algorithm 1-1.

4.2.4.4. Decision variables / outputs

The output of this allocation mechanism is the allocated benefits (or costs) for all community members.

This list of payback / rebilling will then be shared to the stakeholders, which are:

- the community members;
- the energy suppliers of members;
- the distribution system operator (for energy sharing within the same building, to subtract from the bill), and
- the transport system operator (for energy surplus resale out of the community).

4.2.4.5. Fairness

Fairness considerations in benefit allocation will be **implicitly integrated** by evaluating the impact of individual members on the community's overall performance. No explicit fairness parameters are introduced at this stage. The adopted approach is intended to help address several key questions:

- Differentiated energy contracts: When community members have energy contracts at different prices during the same time step, how can fairness be ensured in allocation?
- Equity among local producers: Should a merit order for local production be established, and how can a fair allocation rule be defined? Would introducing a “memory” in the algorithm help ensure that flexible loads are utilised equitably over time?
- Pricing for flexibility: What pricing mechanism should be applied to flexible versus non-flexible consumption, particularly for extended time windows?
- Valuing large flexible consumers: How should higher contributions from consumers with significant flexible capacity relative to their total demand be recognised?
- Valuing distributed small volumes: How can small energy volumes that can be shifted over wide time ranges (i.e., exhibiting high flexibility) be fairly valued?

4.2.5 Algorithm 1-4: Individual portfolio optimisation

Rather than optimising at a community level, this last type of centralised economic dispatch focuses on the individual household level. The OF is therefore modified to reflect the operational preferences of a single member with regard to its assets, instead of looking at operational objectives at a community level.

4.2.5.1. General assumptions and hypotheses

The DER Valorisation Tool is a decision-support framework that helps energy communities and, namely, active consumers optimise the operation of distributed energy resources (DER; solar PV, batteries, EVs, and flexible loads). By performing multi-period optimisation based on demand and renewable forecasts, it determines optimal DER scheduling that maximises renewable self-consumption, minimises curtailment and costs, ensures technical feasibility,

and preserves user comfort. Its scope is to balance economic efficiency, technical feasibility, and sustainability, while unlocking flexibility for energy sharing and market participation.

4.2.5.2. Objective function

The primary objective for the prosumer is to minimise its electricity bill by minimising imports from the grid. The DER's individual objectives (battery cycling, temperature discomfort cost, etc.) are optimised lexicographically in a second step.

Each DER is modelled through the OFs introduced in Section 4.2.1.4.

4.2.5.3. Input data

Time series and assets parameters introduced in Section 4.2.1.2 are available to the community member to perform its individual optimisation.

4.2.5.4. Constraints

The following constraints are needed:

- Balance constraint at the household level, including grid imports and exports.
- Technical operation constraints of all assets owned by the household,
- Contract grid constraints for all contracts linked to the household.

4.2.5.5. Decision variables/ Outputs

- Offtake and injection powers with the outside of the household for the optimisation horizon. These powers can either be trades within the community or with the grid depending on how the member decides to use these quantities.
- Additional metrics, such as self-consumption, self-sufficiency, etc, if relevant.

4.3 Algorithm 2: Centralised market clearing

4.3.1 General formulation and hypotheses

The centralized market clearing represents an auction-based mechanism, which clears orders submitted by community members. Community members are used interchangeably with the term market participant in this section. This algorithm constitutes the bid clearing logic of the auction model presented in [1].

The orders submitted by community members consist of volume-price pairs over multiple timesteps. Volumes correspond to the energy which members are willing to share with / buy from the community, while prices correspond to the value at which they are willing to sell / buy this volume. The order values are determined before running the centralised market clearing algorithm, by solving the Individual portfolio optimisation problem (see section 4.2.5). The volume submitted to the market corresponds to the net volume export/import found by the Individual optimisation problem, while the price submitted to the market corresponds to the marginal benefit of sharing this volume with the community, which is obtained by post-processing the dual variables of the Individual optimisation problem [1].

Volumes not cleared within the centralised market clearing algorithm are passed on to the grid, which either sells or buys these volumes to the individual participants, based on whether the rejected volumes are buys or sells.

4.3.2 Time horizon

The time horizon over which the market clearing algorithms runs can be parametrised by the community. By default, it is set to 24 hours, to represent a whole day of operations. The algorithm can however be rerun several times during a day, to allow community members to adjust their operational schedule based on information updates.

4.3.3 Sets and indices

The sets and indices used in the centralised market clearing algorithm are shown in Table 23.

Table 23: Sets and indices used for the centralised market clearing algorithm

Notation	Description
Ω^J	Set of community members
$\Omega^{I,t,j}$	Set of steps for a given community member at given timestep
$\Omega^{B,j}$	Set of blocks submitted by a given community
j	Community member index j
i	Index of step in order curve
t	Timestep of optimisation horizon
b	Index of block order

For simplicity of representation, a collective asset is modelled in the same way as other assets, meaning it is treated as the sole asset of its corresponding household.

4.3.4 Input data

The fundamental building blocks of the market clearing algorithm are order steps, with a certain price and volume associated to them. This differs from the centralised dispatch algorithms, which are built around the definition of assets with their constraints and their impact on the OF.

These order steps are then aggregated into curves, which can span across one or multiple timesteps. The set of curves over all timesteps considered in the market clearing horizon represent the order of a certain community member.

This hierarchical structure of objects based on orders is outlined in detail in Table 24 to Table 27.

Table 24: Attributes of the OrderStep class

Attribute	Equivalent DTO attribute	Description
$V_{min_{t,i}}^j$	volume_min	Minimum volume which needs to be cleared for the order step to be accepted (this can reflect physical or behavioural operational constraints). If $i > 1$, $V_{min_i}^j = 0$ (only the first step can have a non-zero minimum volume).
$V_{max_{t,i}}^j$	volume_max	Maximum volume which can be cleared for the order step (again, this reflects physical or behavioural operational constraints)
$P_{t,i}^j$	price	Market clearing price

Table 25: Attributes of the BlockOrder class

Attribute	Equivalent DTO attribute	Description
Order ID	order_id	ID of the order
Linked order ID	linked_order_id	ID of linked orders, which allows to identify several orders linked to each other. Linked orders can force the algorithm to follow conditional acceptance rules, depending on their specific design.
Linked block relation (optional)	relation	In linked blocks, blocks can have relational dependencies, such as parent-child, or buy-sell. This parameter specifies which role the considered block has in linked orders.
Block earliest start time (optional)	block_start	This only applies to blocks whose relation is set to “Flexible”. In that case, the start time of the block can be adjusted, provided it is after the earliest start time specified by this parameter.
Block earliest end time (optional)	block_end	This only applies to blocks whose relation is set to “Flexible”. In that case, the end time of the block can be adjusted, provided it is before the latest end time specified by this parameter.
Block timesteps	timesteps	List of timesteps across which the block order is valid.
Order curve	price_volume_curve	List of OrderSteps which constitute the block order curve. For blocks, only one order step (one price-volume pair) is generally defined.

For block orders, the OrderSteps contained in the order curve of block orders are expressed in terms of the index b instead of t .

Table 26: Attributes of the TimestepOrder class

Attribute	Equivalent DTO attribute	Description
Order ID	order_id	ID of the order
Linked order ID	linked_order_id	ID of linked orders, which allows to identify several orders linked to each other. Linked orders can force the algorithm to follow conditional acceptance rules, depending on their specific design.
Timestep	timestep	Timestep for which the order is valid.
Order curve	price_volume_curve	List of OrderSteps which constitute the timestep order curve.

Table 27: Attributes of the Order class

Parameters	Equivalent DTO attribute	Description
Member ID	member_id	Specifies the member to which this order is linked to.
Direction of order (offer or demand)	is_offer	Specifies whether an order is an offer (True) or a demand (False)
List of order curves	timestep_orders	List of order curves submitted by the member within the market clearing horizon. These order curves can be block orders or timestep orders

In addition to these inputs, which contain the information of the orders submitted to the market clearing algorithm, the algorithm also requires a specification of the prioritisation between objectives. For this purpose, an ObjectiveRanking object is created for each objective, and a list of ObjectiveRanking objects is passed to the clearing engine. The input data shown in Table 28 is contained in each ObjectiveRanking object.

Table 28: Attributes of the ObjectiveRanking class

Parameters	Equivalent DTO attribute	Description
objective	objective_name	Objective which should be optimised by the market clearing algorithm
rank	rank	Relative importance
slack	slack	Tolerated deviation (in %) from the optimal result obtained from a higher ranked objective, once optimising for the considered objective.
direction	direction	Specifies whether the objective should be minimised or maximised.

Finally, a pricing model must be specified to apply the correct pricing rule to the orders cleared in the market. For this, different pricing rules are predefined, which are imported as ClearingPriceModel classes.

Pay-as-bid price model (ClearingPriceModel):

For a pay-as-bid price model, no parameters to provide as input, as orders are cleared at their order price submitted to the market.

For a marginal price-based model, the attributes shown in Table 29 are used.

Table 29: Attributes used for ClearingPriceModel implementing a marginal price model

Parameters	Equivalent DTO attribute	Description
Offer weight	offer_weight	Parameter which assigns a weight (between 0 and 1) to the marginal sell price, in order to calculate a clearing price which is a weighted average of the marginal sell and marginal buy price.

4.3.4.1. Variables

$a_{t,i}^j$ (Boolean): Acceptance of order steps for timestep orders.

$r_{t,i}^j$ (float): Volume acceptance ratio of accepted order steps for timestep orders

$a_{b,i}^j$ (Boolean): Acceptance of order steps for block orders.

$r_{b,i}^j$ (float): Volume acceptance ratio of accepted order steps for block orders

4.3.5 Objective

The following objectives can be considered by the market clearing algorithm:

- maximisation of social welfare;
- minimisation of community bill (*cost – revenue*);
- minimisation of uncleared bid volumes;
- maximisation of local Renewable Energy (RE) consumption (equivalent to minimisation of grid imports).

Several of these objectives can be included in the market clearing algorithm, depending on the community's preferences. These objectives are then cleared lexicographically, meaning that they are cleared sequentially, taking the result of the previous optimisation as an additional upper / lower-bound constraint in the next optimisation. This additional constraint can be relaxed by adding a slack variable, which tolerates a deviation from the previous optimum to improve the optimal solution in the current problem, effectively allowing a trade-off across multiple objectives.

4.3.6 Outputs

Outputs:

Depending on the chosen design, orders are not necessarily cleared at a single market price. Bid processing could lead to different prices amongst participants.

- Cleared order by market participant (offer or demand): $C_t^j(V_{cleared,t}^j, P_{cleared,t}^j)$:

- $V_{cleared,t}^j$ (float): cleared volume;
- $P_{cleared,t}^j$ (float): market clearing price.

- Market clearing statistics:

- Social welfare (float);
- Total volume cleared (float);

- Total RE volume consumed (float);
- Community bill (float).

4.4 Algorithm 3: Pricing mechanism

The pricing model algorithm aims to compute internal pricing within the community. When executed with forecast profiles, it gives a price signal for community members to adapt their consumption and production. When executed with the actual profiles, it gives ex-post the prices that will be used for the settlement phase.

4.4.1 General assumptions

Every household signs the same contract with one energy supplier to facilitate internal price determination. The contract category does not affect the mechanism.

Each household is equipped with a HEMS, which helps optimise its energy usage.

The CM is responsible for managing all market transactions and determining internal prices. This role may be fulfilled by an algorithm-driven virtual platform or a traditional intermediary.

4.4.2 Pricing model of the community manager

The CM calculates internal prices ex-post using the following inputs and mechanisms:

4.4.2.1. Input parameters

Table 30 presents the input parameters (also referred to as attributes when taking an object-oriented perspective) necessary to run the pricing mechanism algorithm.

Table 30: Attributes of the PricingMechanism class used to calculate internal prices

<i>Parameter</i>	<i>Description</i>
Imp_t	Aggregated community import profile
Exp_t	Aggregated community export profile
λ_t^{imp}	Retail import prices
λ_t^{exp}	Retail export prices
Pricing rule	Predefined pricing rule

4.4.2.2. Pricing mechanisms

In theory, many different pricing mechanisms can be implemented based on the input parameters provided above, by changing the relative weight of consumers and producers within the community.

Based on the status of Deliverable 2.3. [1], mechanisms which are currently considered for implementation are:

- a mid-market rate, which calculates the weighted average of the retail import and export prices;
- a price based on the local supply-demand balance ratio;

Further details on these pricing mechanisms are provided in [1], but are not important at this stage to implement the DTOs of this type of algorithm.

4.4.2.3. Output

- Internal prices (λ_t^{buy} & λ_t^{sell}) published by the CM ex-post;
- Other metrics at the community level, such as collective self-consumption and fairness.

4.5 Algorithm 4: Heuristic benefit allocation

Benefit allocation mechanisms introduced in Section 4.2.4 rely on an optimisation approach based on mathematical programming. Although more efficient, the outcomes of such mechanisms can be difficult to interpret for community members, effectively reducing the transparency of the benefit allocation mechanism. Hence, they are compared to more practical ones that are based on injection/ offtake profiles of community members.

4.5.1 Solving methods

Several heuristic approaches are considered to tackle a fair and efficient benefit allocation. They are listed below and further described in Deliverable D.2.3 [1].

- **Fixed sharing keys:** the allocation is determined using fixed sharing keys between members. Those keys can be applied to the aggregated injected/ withdrawn energy or directly to the total associated cost;
- **Pro-rate consumption:** the allocation is proportional to the member's total consumption;

- **Hybrid solution:** the allocation is based on a mix of the 'Fixed sharing keys' and 'Pro-rate consumption' approaches;
- Multi-round of fixed sharing keys:** Iteratively allocating shared energy based on pre-defined sharing keys.

4.5.2 Input/ Output

The different approaches require the following inputs:

- List of sharing keys (keys must be non-negative and sum to 1), if relevant;
- Offtake and injection profiles from each community member;
- The total benefit of the community.

All approaches return the benefit redistribution for each community member as:

- Start time of benefits redistribution horizon;
- End time of benefits redistribution horizon;
- Financial value of benefits redistribution.

5 Data schemes

The data schemes for the algorithms presented above outline the input and output formats of each algorithm. These data schemes are structured around DTO, consisting of classes with attributes and dependencies between classes. Each class represents either physical elements of the EC (asset, community member) or conceptual elements (bids, objectives).

These DTOs are defined using the Pydantic library in Python, from which an API is built using FastAPI following OpenAPI specifications. For this deliverable, the content of the API, which currently runs on a local server only, is shared through the html document “u2demo-optimization-doc.html”, which is hosted on the deliverable Github repository in the U2Demo project: <https://github.com/U2DemoProject/AlgorithmDTOsWP4T1>.

For readability purposes, one mermaid file is also provided for each algorithm (script provided in APPENDIX B – Data Transfer Objects for Algorithms). Each of these mermaid files is summarised below using charts.

Each Pydantic class is represented in Figure 21 and all subsequent figures by a blue rectangle. Inheritance links are shown as arrows pointing from the Child class to the Parent class. The “Commons” library the classes used across all algorithms. This concerns mainly community components found across all algorithms, such as Community Members, Suppliers, Meters, Contracts, as well as standard objects such as TimeValue, Prices, Forecasts, Schedules.

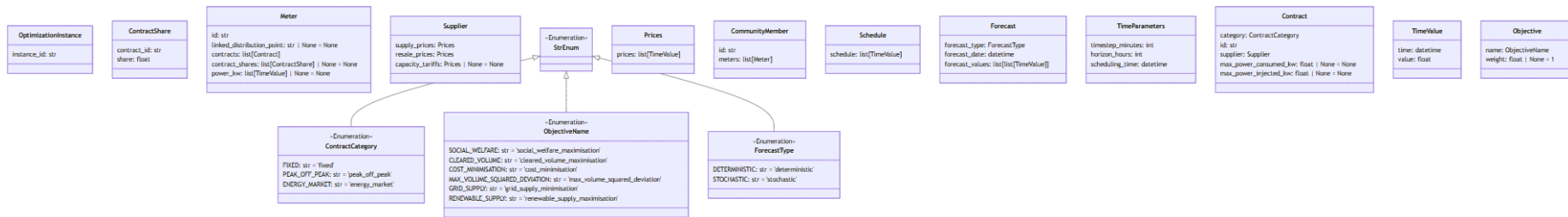


Figure 21: DTO for classes in Commons file

Figure 22 shows the classes used across all energy management system algorithms. This mainly includes all the standard assets found in an EC, such as PVs, EVs, heat pumps, etc. There is a strong level of inheritance between these classes, in order to reuse concepts across classes.

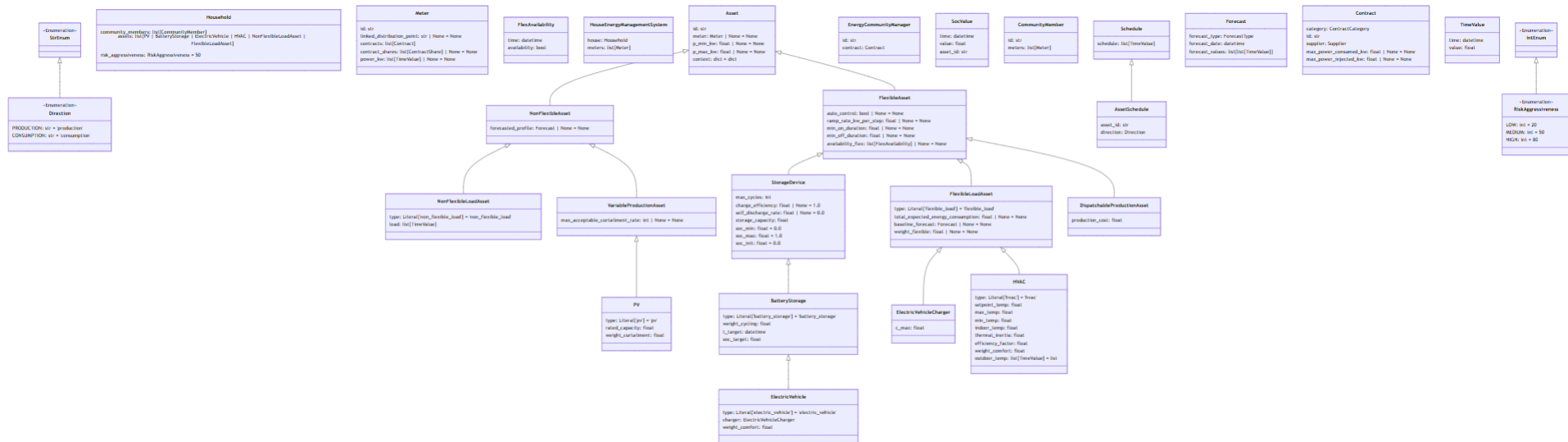


Figure 22: DTO for classes in Commons centralised energy management system

Figure 23 shows the elements which compose a centralised energy management system at the community level. It is fact mainly the list of Households which contains the information regarding all the assets to schedule optimally. These assets follow the structure defined in Figure 22. Each asset is linked to a Meter, and each Meter is linked to a list of Contracts, allowing an asset to be shared by multiple community members. Conversely, each Member can have several Meters, with different Contracts and therefore different Suppliers. This allows to model a wide range of business models within the EC, although all are scheduled centrally.

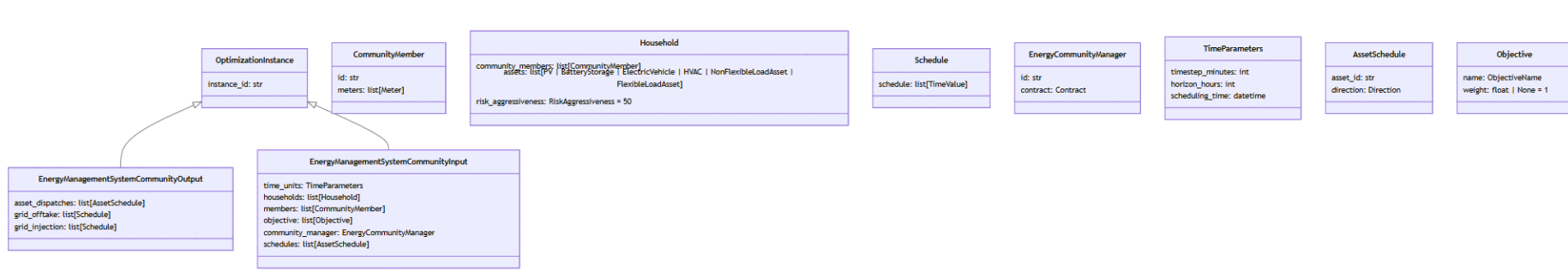


Figure 23: DTO for classes in Algorithm 1-1 (Centralised energy management system for Community Energy Management System)

Figure 24 shows the centralised energy management system with the P2P context. Here, rather than Households with their corresponding assets, the basic building blocks are Peers, with their net position, cost function, and production differentiation parcel for each of the other Peers.

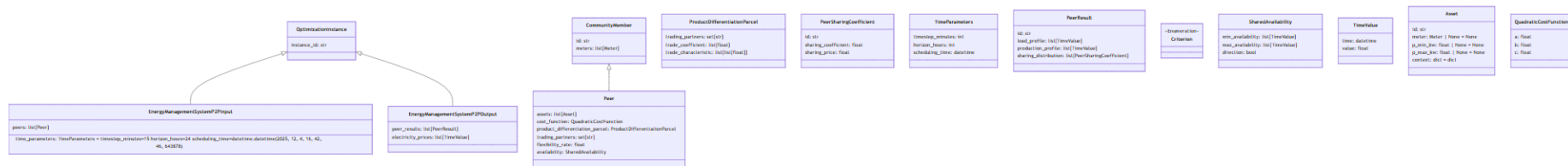


Figure 24: DTO for classes in Algorithm 1-2 (Centralised energy management system with P2P preferences)

Figure 25 shows the Centralised Energy Management System for Collective Benefit Allocation. In addition to the assets with their corresponding constraints, this algorithm also takes the assets' dispatch schedule as an input, as it runs ex-post and calculates the optimal redistribution of benefits across community members.

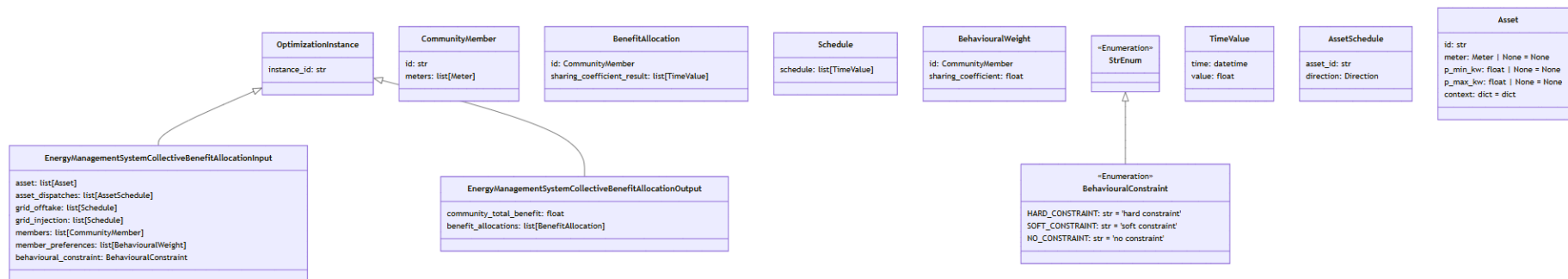


Figure 25: DTO for classes in Algorithm 1-3 (Centralised energy management system for Collective benefit allocation (settlement))

In Figure 26, the Individual Portfolio Optimisation algorithm is shown, which only considers one Household instead of a list of Households as input. In addition, the objectives with their level of prioritisation at the household must also be specified as inputs. The output is the optimal operational schedule of assets at a household level, based on the desired objectives.

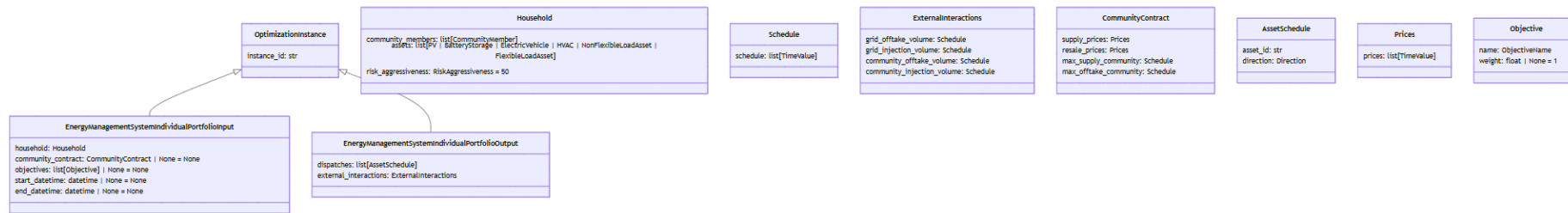


Figure 26: Centralised Energy Management System for Individual Portfolio Optimisation

In Figure 27, the centralised market clearing algorithm is shown. In this case, the fundamental building blocks are TimeStepOrders (for orders over a single timestep) and BlockOrders (for orders spanning multiple timesteps). These volume-price steps are aggregated for each timestep / block to create OrderCurves, which are sent to the clearing algorithm. The clearing algorithm, which clears objectives lexicographically, needs to have this ranking of objectives specified.

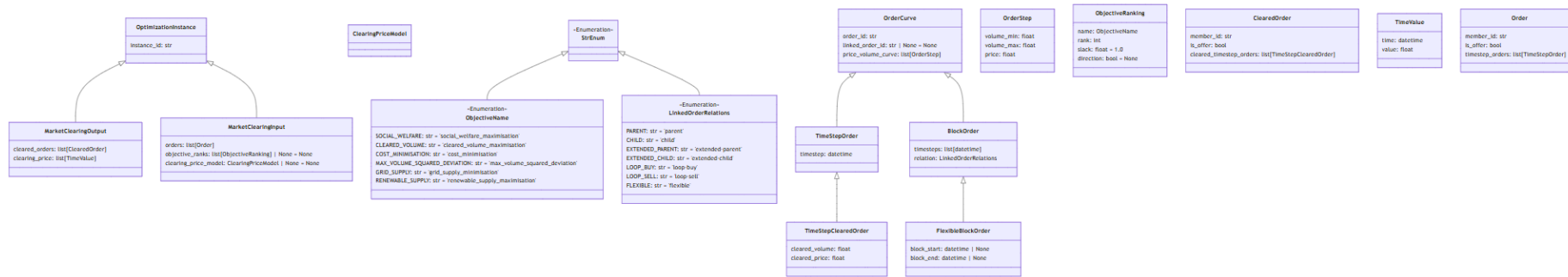


Figure 27: DTO for classes in Algorithm 2 (Centralised market clearing)

Figure 28 shows the classes implemented for the pricing mechanism algorithms. In this case, the main inputs are the time series of loads and productions (list of TimeValues) and the pricing rule chosen, which is an Enum activated a pre-defined pricing logic.

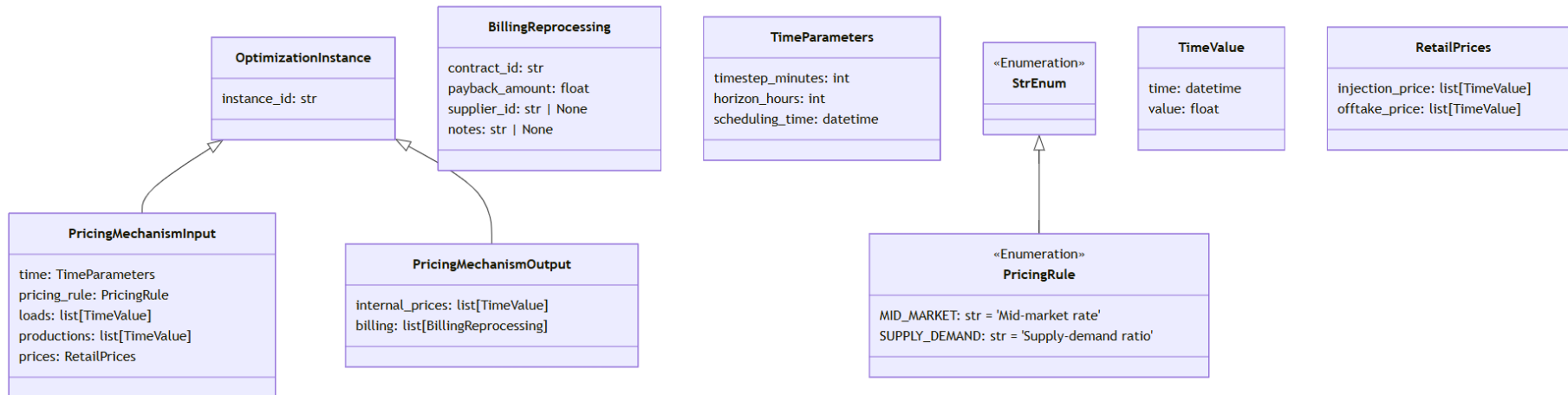


Figure 28: DTO of classes for Algorithm 3: Pricing mechanism

Figure 29 shows the heuristic benefit allocation, which imports the list of operational schedules for the members (as it runs ex-post) and the predefined ContractShare for each member. The corresponding output is the CommunityMemberBenefit which each member obtains from the benefit allocation process.

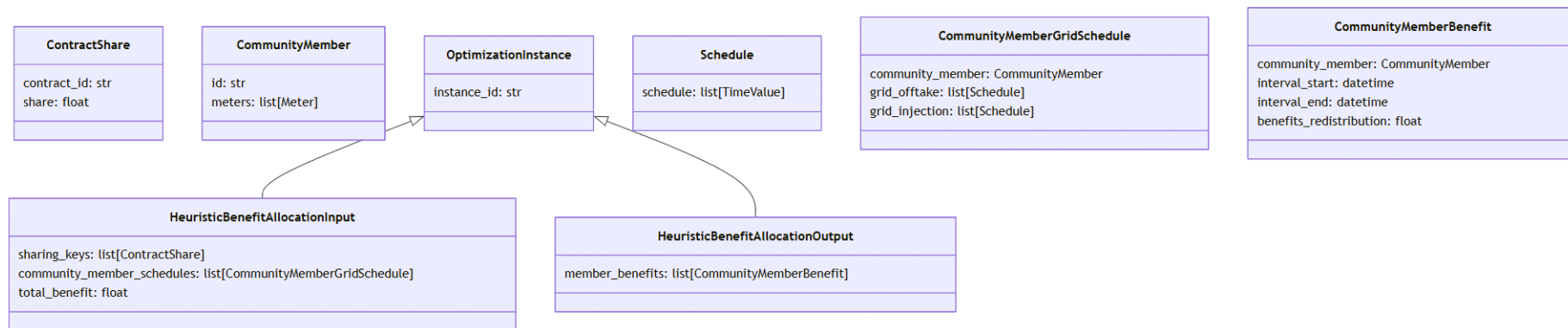


Figure 29: DTO of classes for Algorithm 4: Heuristic benefit allocation

6 Technical architecture

6.1 Overview

This document describes the architecture and operation of an optimisation engine developed as a standalone microservice. The microservice will be deployed via a Docker container and exposed through a messaging-based interface. Its primary function is to receive, process, and return results of optimisation problems for various energy-related operations.

6.2 Deployment and Configuration

- **Containerization:** The service will be packaged and deployed as a Docker image.
- **Configuration:** Parametrisation will be managed using environment variables, allowing flexible deployment across environments.
- **Resource Allocation:** CPU and memory requirements will depend on the complexity of the optimisation problem and will be defined in coordination with system administrators or DevOps engineers at a later stage.

6.3 Communication Architecture

The microservice communicates with other components via a **messaging system based on AMQP** (Advanced Message Queuing Protocol), such as **RabbitMQ**, an open-source message broker.

Beyond its proven benefits in terms of fault tolerance and smooth real-time performance, RabbitMQ is of particular interest in the U2Demo project due to its handling of channels, which allow to have multiple communication paths without suffering from excessive overheads [8]. This is important not only due to the multiple households / assets for which some of the algorithms are implemented simultaneously, but also to run the same algorithm with different parameter configurations. This allows to implement sensitivity studies or iterative approaches, if relevant.

RabbitMQ is a queue-based messaging model, involving a publisher (the U2Demo platform), an exchange (defined in the message broker) routing the messages to the appropriate queue, queues (defined in the message broker) storing messages for the appropriate consumer, and the consumer (the docker image subscribed to that queue and containing the algorithm which runs based on the information sent by the message).

The system is designed around **dedicated exchanges**, each corresponding to a specific operation:

- centralized-dispatch;
- market-clearing;
- pricing-model;
- heuristic-benefit-allocation;

Each exchange handles incoming optimisation tasks for its respective domain. The queue type and the exchange type to implement (direct, fanout, topic, headers exchange) must be chosen in collaboration with the partners developing the U2Demo platform.

An overview of the communication architecture is provided in Figure 30 (indicative figure, changes are expected in further tasks):

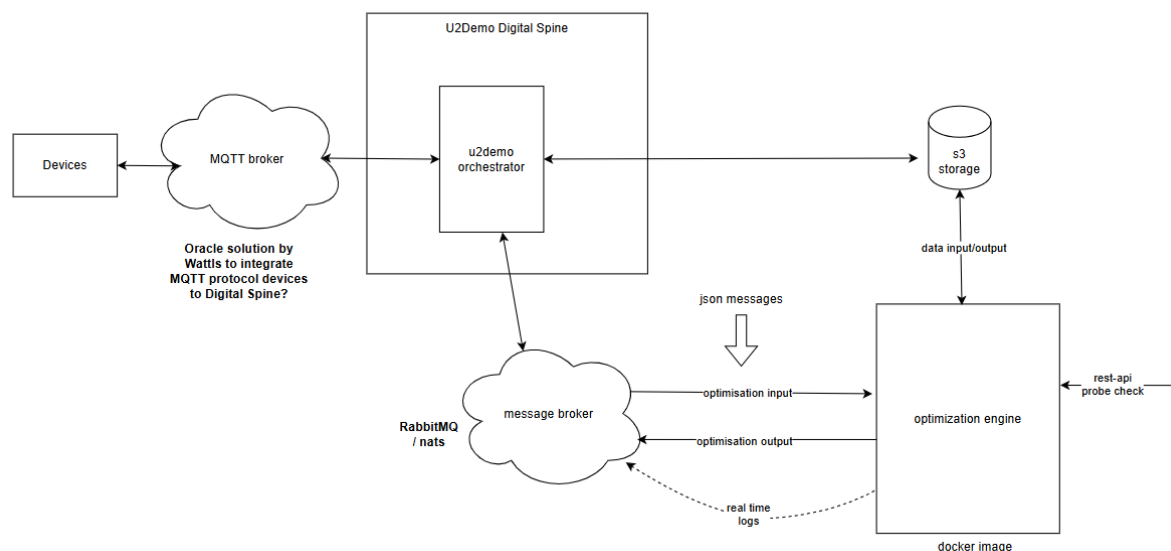


Figure 30: Communication architecture between the optimisation engine and U2Demo orchestrator

Within the Digital Spine, the u2demo orchestrator manages the overall workflow and routes JSON messages to the message broker (RabbitMQ/NATS). The optimisation engine receives input instances from the message broker, performs the requested computations, and returns optimisation results back through the same messaging infrastructure. Throughout execution, the engine stores data in S3 storage for input/output persistence and streams real-time logs back to the message broker for monitoring purposes. The engine can also be probed via a REST API for health checks and status queries. All components operate as containerised Docker images, ensuring portability and consistent deployment across environments.

6.4 Message format

6.4.1 Input Data Structure

Optimisation requests are submitted as **AMQP messages**. The **message body** contains the full definition of the optimisation problem, referred to as the **input instance**:

- Each input instance is assigned a **unique identifier (UUID)**;
- All required data should be included in the message body, following the format specified through the DTOs in section 5 and the corresponding Github repository;
- If the data payload is too large, the message can include **presigned public URLs** (e.g., from S3 or equivalent) to download additional input files.

6.4.2 Output & Results

Upon completion, the microservice will publish the **solution output** to a corresponding **output exchange**, referencing the **same unique input ID** for traceability:

- The output message will include status, objective values, and any relevant result data;
- Consumers can match responses with requests using the instance ID.

6.5 Real-Time Logging and Debugging

Logging is handled on two levels:

- **Real-time progress logs** will be streamed to a **dedicated AMQP exchange**, enabling live monitoring of ongoing computations;
- **Detailed debug logs** will be saved in a **mounted volume** on disk. Each optimisation instance will have its own directory, named after its unique ID, for organised access and post-mortem analysis.

7 Conclusions

7.1 Summary

The aim of Deliverable D4.1 was to identify and describe the energy algorithms to develop in the rest of Work Package (WP) 4. The identification of relevant algorithms consisted of comparing and abstracting the energy sharing activities and models studied in previous WPs, to propose algorithms which are reusable enough to be implemented in different contexts (adaptability), while capturing the specificities of each demonstration site (suitability). This abstraction work allowed us to identify commonalities and dependencies between energy sharing models, which subsequently allowed to propose a set of algorithms with a standardised data format.

The algorithm types to be implemented are grouped into four types:

- Energy management system algorithms, with a central agent setting the operational profile of the optimised assets and/or households, based on constraints and objectives defined by the community or imposed by the assets;
- Centralised market clearing algorithms, with an auction-based clearing mechanism receiving orders from community members and clearing them based on the objectives and constraints defined by the community;
- Pricing mechanism algorithms, which calculate different EC prices that community members react to, indirectly benefiting the community at the same time;
- Heuristic benefit allocations algorithms, which redistribute the total financial benefits of position netting within the energy communities, and adjust the energy/capacity allocation among the members accordingly. This redistribution is based on sharing coefficients decided by community members.

For each of these algorithm types, specifications were provided to outline the data formats of the inputs and outputs to be expected from these algorithms. In the case of centralised economic dispatch algorithms, the formalisation could be pushed one step further by predefining assets with their constraints and objectives, which can then be easily instantiated in any specific economic dispatch problem.

These specifications were then formalised for each algorithm using DTOs, expressed as classes with attributes of a specific type and, whenever relevant, predefined methods. Common attributes shared across multiple assets were defined in parent classes, which specific assets can inherit from. This object-oriented approach provides a modular structure to the algorithms, allowing them to be easily adapted to different contexts while reusing the same fundamental building blocks. This is particularly important in the local EC context, where solutions must be tailored to the specific needs and properties of the community, while the small scale of the problem makes ad hoc developments infeasible.

The framework provided by deliverable 4.1 thereby ensures that the algorithms developed in the rest of WP4 remain both adap and reusable.

7.2 Main Challenges

The main challenge of this deliverable is its integration with existing and upcoming work in the U2Demo project, both regarding information structure coordination and technological coordination.

- Regarding the information structure, the DTOs defined in this report aim to formalise the structure of the inputs provided to each of the algorithms, in order to have a clear communication protocol with the U2Demo platform. However, these inputs are combined from different sources. For example, some assets require inputs from the forecasting service in case forecast are needed, from the meter linked to the asset in case historical measurements are needed, from the user if behavioural preference parameters are needed, and from the asset itself to include technical specifications.

This diversity of sources implies that a data processing step is required within the U2Demo platform to make the data collected from the demo sites compatible with the algorithm inputs. Conversely, the outputs from the algorithms need to be post-processed in order to send the information back to the demo sites. In further work, the EC ontology developed in deliverable D3.2 and the EC algorithms ontology developed in this deliverable must therefore be further connected to ensure coherent and efficient data communication throughout the platform.

Finally, the challenge of information synthesis in this deliverable is compounded by the fact that it depends on still ongoing tasks (e.g. T2.3, T2.3, and T4.2-T4.3). The reporting might therefore not exactly reflect the final structure that will be implemented after those tasks are complete.

- Regarding the technological coordination, the choice of message broker technology (RabbitMQ) needs to align with the implementation of the U2Demo platform. Moreover, integration developments need to link the exchanges and queues of the message broker to the corresponding docker images of each algorithm.

7.3 Next deliverables

Deliverable D4.1 establishes the energy sharing ontology to be used when implementing the energy sharing algorithms in the rest of WP4. The algorithms developed in T4.3, which are the deliverables of that task, will therefore aim to combine the formalism defined in this deliverable with the conceptual and mathematical formulation developed in WP2.

Moreover, the flexibility service algorithms developed in T4.4, which are based on the proof-of-concept work in WP2, will also benefit from the formalism defined in this deliverable, as they are strongly based on the . Finally, T4.2, which focuses on the further development and implementation of the individual decision-making models and forecasting algorithms developed in [3], will also benefit from the ontology defined in deliverable D4.1 in the following ways:

- the inputs, outputs, constraints, and OFs for the individual decision-making models have been defined in this deliverable (see section 4.2.5), providing the description of what needs to be implemented;
- the output format of the forecast algorithms to develop in T4.2 has been specified by defining the forecast objects which the different algorithms need as inputs.

Overall, deliverable D4.1 provides the foundation for the rest of the work in WP 4, by bridging the proof-of-concept work in WP2 with the higher TRL development in WP4.

However, this work must also be taken into account to the platform development work in WP 3, to ensure that the data flows are translated properly between the demo sites and the algorithms, and that the algorithms are properly integrated into the platform through an appropriate message broker setup.

Finally, the algorithms defined in this deliverable are also essential for the work in WP5, as they define the set of possible test cases which can be implemented in the demo sites based on the developed energy sharing algorithms. The work in this deliverable must therefore be communicated further to each demo site, to allow them to identify the most suitable test case setup.

8 References

- [1] E. Delarue *et al.*, “Deliverable D 2.3 - P2P trading matching and Energy Sharing models,” Use of open-source P2P energy sharing platforms for energy democratization (U2Demo) Project, ref 101160684, 2025.
- [2] NASA, “Technology Readiness Level Definitions.” NASA, 2017. Accessed: Oct. 21, 2025. [Online]. Available: https://www.nasa.gov/wp-content/uploads/2017/12/458490main_trl_definitions.pdf
- [3] I. Laouali *et al.*, “U2Demo Deliverable D 2.2 Decision support methods for active consumers and ECs,” Use of open-source P2P energy sharing platforms for energy democratization (U2Demo) Project, grant agreement 101160684, 2025.
- [4] M. A. K. Bodelier *et al.*, “Deliverable D1.2 - Active consumers’ needs and social parameters,” Use of open-source P2P energy sharing platforms for energy democratization (U2Demo) Project, grant agreement 101160684, 2025.
- [5] V. Deichmann *et al.*, “Deliverable D1.4 - U2Demo Use Case Repository,” Use of open-source P2P trading and energy sharing (U2Demo) Horizon Europe funded project, grant agreement 101160684, 2025.
- [6] E. Baeyens, E. Y. Bitar, P. P. Khargonekar, and K. Poolla, “Coalitional Aggregation of Wind Power,” *IEEE Trans. Power Syst.*, vol. 28, no. 4, pp. 3774–3784, Nov. 2013, doi: 10.1109/TPWRS.2013.2262502.
- [7] L. Han, T. Morstyn, and M. McCulloch, “Incentivizing prosumer coalitions with energy management using cooperative game theory,” in *IEEE Transactions on Power Systems* 34 (1), 2019. doi: 10.1109/TPWRS.2018.2858540.
- [8] E. Zanetti, “RabbitMQ: A Complete Guide to Message Broker, Performance, and Reliability.” Medium, 2025. Accessed: Oct. 29, 2025. [Online]. Available: <https://medium.com/@erickzanetti/rabbitmq-a-complete-guide-to-message-broker-performance-and-reliability-3999ee776d85>

APPENDIX A: Algorithm 1-1 – Multi-objective example

Here, a multi-objective example for Algorithm 1.1 is provided.

- Baseline – EV charge as soon as possible and direct self-consumption (result expressed in kWh):

$$\min F_1 = \sum_{ev \in EV} \sum_{t \in T} \frac{E_{ev}^{maxEV} - E_{ev,t}^{EV}}{\Delta t} + P_t^{imp} \quad (0-1)$$

where E_{ev}^{maxEV} represents the EV battery maximum capacity, $E_{ev,t}^{EV}$ represents the EV current SoC, P_t^{imp} represents the grid import power and Δt represents the optimisation timestep. In this section, it is considered that the timestep is one hour.

- Reduce operational cost (result expressed in €) – reduce the total cost considering energy invoice costs and the assets costs (e.g. charging and discharging costs for EVs and BESS, generator maintenance and load services activation costs):

$$\begin{aligned} \min F_2 = & \sum_{t \in T} P_t^{imp} \cdot \lambda_t^{imp} - P_t^{exp} \cdot \lambda_t^{exp} + \sum_{b \in B} \sum_{t \in T} P_{b,t}^{chBESS} \cdot \lambda_{i,b,t}^{chBESS} + P_{b,t}^{dchBESS} \\ & \cdot \lambda_{b,t}^{dchBESS} + \sum_{ev \in EV} \sum_{t \in T} P_{ev,t}^{chEV} \cdot \lambda_{ev,t}^{chEV} + P_{ev,t}^{dchEV} \cdot \lambda_{ev,t}^{dchEV} \\ & + \sum_{l \in L} \sum_{t \in T} P_{l,t}^{loadRED} \cdot \lambda_{l,t}^{loadRED} + P_{l,t}^{loadCUT} \cdot \lambda_{l,t}^{loadCUT} \\ & + \sum_{g \in G} \sum_{t \in T} P_{g,t}^{genAct} \cdot \lambda_{g,t}^{genAct} + P_{g,t}^{genExc} \cdot \lambda_{g,t}^{genExc} \end{aligned}$$

- P_t^{imp} and P_t^{exp} are the grid import and export energy at time t respectively, with λ_t^{imp} and λ_t^{exp} representing their corresponding cost.
- $P_{g,t}^{genAct}$ is the energy used by the community at time t , with $\lambda_{g,t}^{genAct}$ representing its operating or opportunity cost for each generator $g \in G$.
- $P_{g,t}^{genExc}$ is the surplus energy that is not utilised internally (spilled or curtailed), and $\lambda_{g,t}^{genExc}$ is the corresponding penalty or opportunity cost for each generator $g \in G$.
- $P_{l,t}^{loadRED}$ represents voluntary demand reduction relative to the baseline schedule at time t , and $\lambda_{l,t}^{loadRED}$ is the cost associated with that reduction for each load $l \in L$,
- $P_{l,t}^{loadCUT}$ represents voluntary curtailment at time t , and $\lambda_{l,t}^{loadRED}$ is the corresponding cost for each load $l \in L$,

- $P_{b,t}^{chBESS}$ and $P_{b,t}^{dchBESS}$ are the energy charged and discharged, with $\lambda_{b,t}^{chBESS}$ and $\lambda_{b,t}^{dchBESS}$ their respective marginal costs (e.g., cycling, efficiency losses) for each BESS unit $b \in B$,
- $P_{ev,t}^{chEV}$ and $P_{ev,t}^{dchEV}$ are the energy charged and discharged with $\lambda_{ev,t}^{chEV}$ and $\lambda_{ev,t}^{dchEV}$ being the corresponding costs for each EV $ev \in EV$.

- Reduce energy invoice – minimise electricity costs (result expressed in €):

$$\min F_3 = \sum_{t \in T} P_t^{imp} \cdot \lambda_t^{imp} - P_t^{exp} \cdot \lambda_t^{exp}$$

- Reduce grid import (result expressed in kWh):

$$\min F_4 = \sum_{t \in T} P_t^{imp}$$

- Increase battery longevity – preserve battery's lifespan (result expressed in kWh):

$$\min F_5 = \sum_{b \in B} \sum_{t \in T} P_{b,t}^{chBESS}$$

- Reduce environmental impact – reduce CO₂ emissions (result expressed in kg_{CO2} kWh):

$$\min F_6 = \sum_{t \in T} P_t^{imp} \cdot EF_t$$

- EF_t represents the emission factor in kg CO₂/kWh.
- Increase comfort – Prioritise member routines: no delay or restriction on appliance operation (result expressed in kWh):

$$\min F_7 = \sum_{l \in L} \sum_{t \in T} P_{l,t}^{loadCUT} + P_{l,t}^{loadRED} + P_{l,t}^{loadENS}$$

- $P_{l,t}^{loadRED}$ represents voluntary demand reduction relative to the baseline schedule at time t for each load $l \in L$,
- $P_{l,t}^{loadCUT}$ represents voluntary curtailment at time t for each load $l \in L$,
- $P_{l,t}^{loadENS}$ represents power not supplied by involuntary curtailment at time t for each load $l \in L$.

An EC member may also pursue multiple objectives simultaneously. Then, these OFs can be combined into a multi-objective function. In the proposed approach, this is done by assigning weights to the different OFs, thereby reflecting their relative importance. To this end, the multi-objective function can be formulated as follows:

$$\min F = \sum_{i \in I} \sum_{n \in N} \alpha_{i,n} \cdot \beta_{i,n} \cdot F_{i,n} \quad (0-2)$$

where $i \in I$ refers to the EC members and $n \in N$ refers to OF. The weights $\alpha_n \in [0,1]$ set the contribution of each objective: when an objective is deactivated, $\alpha_n = 0$; when only one objective is considered, $\alpha_n = 1$ and the others are zero; in the multi-objective case, the weights satisfy $\sum_{n \in N} \alpha_n = 1$. The scaling factors $\beta_n > 0$ normalize the objectives, ensuring comparable magnitudes. Table 31 shows the OF correlation with β_n factor.

Table 31: OF correlation with β_n factor

OF	OF result unit	β_n	β_n Unit
Baseline	kWh	$\overline{C_{imp}}$	€/kWh
Reduce operational cost	€	1	–
Reduce energy invoice	€	1	–
Reduce Grid Import	kWh	$\frac{\overline{C_{imp}} + \overline{C_{exp}}}{2}$	€/kWh
Battery Longevity	kWh	$\overline{C_{imp}}$	€/kWh
Environmental Impact	Kg _{CO2} kWh	$\frac{\overline{C_{imp}}}{\overline{EF}}$	€/kWh · 1/(Kg _{CO2})
Comfort	kWh	$\overline{C_{imp}}$	€/kWh

In Table 31, $\overline{C_{imp}}$ represents the average import price, $\overline{C_{exp}}$ represents the average export price and \overline{EF} that represents the average emission factors.

APPENDIX B – Data Transfer Objects for Algorithms

The DTOs for each algorithm are detailed below using Mermaid files for readability purposes. The corresponding JSON, HTML and source code in Python are found in the U2Demo Github repository: <https://github.com/U2DemoProject/AlgorithmDTOsWP4T1>.

B.1 Commons DTO

```
```mermaid
classDiagram
 class OptimizationInstance {
 instance_id: str
 }
 class ContractShare {
 contract_id: str
 share: float
 }
 class Meter {
 id: str
 linked_distribution_point: str | None = None
 contracts: list[Contract]
 contract_shares: list[ContractShare] | None = None
 power_kw: list[TimeValue] | None = None
 }
 class ContractCategory {
 <<Enumeration>>
 FIXED: str = 'fixed'
 }
```

```
 PEAK_OFF_PEAK: str = 'peak_off_peak'
 ENERGY_MARKET: str = 'energy_market'
}

class Supplier {
 supply_prices: Prices
 resale_prices: Prices
 capacity_tariffs: Prices | None = None
}

class ForecastType {
 <<Enumeration>>
 DETERMINISTIC: str = 'deterministic'
 STOCHASTIC: str = 'stochastic'
}

class StrEnum {
 <<Enumeration>>
}

class Prices {
 prices: list[TimeValue]
}

class CommunityMember {
 id: str
 meters: list[Meter]
}

class Schedule {
 schedule: list[TimeValue]
```

```
}

class ObjectiveName {
 <<Enumeration>>
 SOCIAL_WELFARE: str = 'social_welfare_maximisation'
 CLEARED_VOLUME: str = 'cleared_volume_maximisation'
 COST_MINIMISATION: str = 'cost_minimisation'
 MAX_VOLUME_SQUARED_DEVIATION: str = 'max_volume_squared_deviation'
 GRID_SUPPLY: str = 'grid_supply_minimisation'
 RENEWABLE_SUPPLY: str = 'renewable_supply_maximisation'
}

class Forecast {
 forecast_type: ForecastType
 forecast_date: datetime
 forecast_values: list[list[TimeValue]]
}

class TimeParameters {
 timestep_minutes: int
 horizon_hours: int
 scheduling_time: datetime
}

class Contract {
 category: ContractCategory
 id: str
 supplier: Supplier
 max_power_consumed_kw: float | None = None
 max_power_injected_kw: float | None = None
}
```

```
class TimeValue {
 time: datetime
 value: float
}

class Objective {
 name: ObjectiveName
 weight: float | None = 1
}

StrEnum <|-- ContractCategory
StrEnum <|-- ObjectiveName
StrEnum <|-- ForecastType
```

```
...
```

## B.2 Commons energy management system DTO

---

```
```mermaid
classDiagram
    class NonFlexibleLoadAsset {
        type: Literal['non_flexible_load'] = 'non_flexible_load'
        load: list[TimeValue]
    }

    class IntEnum {
        <<Enumeration>>
    }
```

```
class DispatchableProductionAsset {
    production_cost: float
}

class Household {
    community_members: list[CommunityMember]
    assets: list[PV | BatteryStorage | ElectricVehicle | HVAC |
NonFlexibleLoadAsset | FlexibleLoadAsset]
    risk_aggressiveness: RiskAggressiveness = 50
}

class Meter {
    id: str
    linked_distribution_point: str | None = None
    contracts: list[Contract]
    contract_shares: list[ContractShare] | None = None
    power_kw: list[TimeValue] | None = None
}

class FlexAvailability {
    time: datetime
    availability: bool
}

class NonFlexibleAsset {
    forecasted_profile: Forecast | None = None
}

class Direction {
    <<Enumeration>>
    PRODUCTION: str = 'production'
```

```
    CONSUMPTION: str = 'consumption'
}

class FlexibleAsset {
    auto_control: bool | None = None
    ramp_rate_kw_per_step: float | None = None
    min_on_duration: float | None = None
    min_off_duration: float | None = None
    availability_flex: list[FlexAvailability] | None = None
}

class ElectricVehicleCharger {
    c_max: float
}

class PV {
    type: Literal['pv'] = 'pv'
    rated_capacity: float
    weight_curtailment: float
}

class HouseEnergyManagementSystem {
    house: Household
    meters: list[Meter]
}

class StrEnum {
    <<Enumeration>>
}

class EnergyCommunityManager {
```

```
    id: str
    contract: Contract
}

class AssetSchedule {
    asset_id: str
    direction: Direction
}

class StorageDevice {
    max_cycles: int
    charge_efficiency: float | None = 1.0
    self_discharge_rate: float | None = 0.0
    storage_capacity: float
    soc_min: float = 0.0
    soc_max: float = 1.0
    soc_init: float = 0.0
}

class SocValue {
    time: datetime
    value: float
    asset_id: str
}

class CommunityMember {
    id: str
    meters: list[Meter]
}

class Schedule {
```

```
    schedule: list[TimeValue]
  }

class Forecast {
  forecast_type: ForecastType
  forecast_date: datetime
  forecast_values: list[list[TimeValue]]
}

class BatteryStorage {
  type: Literal['battery_storage'] = 'battery_storage'
  weight_cycling: float
  t_target: datetime
  soc_target: float
}

class Contract {
  category: ContractCategory
  id: str
  supplier: Supplier
  max_power_consumed_kw: float | None = None
  max_power_injected_kw: float | None = None
}

class RiskAggressiveness {
  <<Enumeration>>
  LOW: int = 20
  MEDIUM: int = 50
  HIGH: int = 80
}
```

```
class HVAC {
    type: Literal['hvac'] = 'hvac'
    setpoint_temp: float
    max_temp: float
    min_temp: float
    indoor_temp: float
    thermal_inertia: float
    efficiency_factor: float
    weight_comfort: float
    outdoor_temp: list[TimeValue] = list
}

class TimeValue {
    time: datetime
    value: float
}

class FlexibleLoadAsset {
    type: Literal['flexible_load'] = 'flexible_load'
    total_expected_energy_consumption: float | None = None
    baseline_forecast: Forecast | None = None
    weight_flexible: float | None = None
}

class Asset {
    id: str
    meter: Meter | None = None
    p_min_kw: float | None = None
    p_max_kw: float | None = None
    context: dict = dict
}
```

```
class ElectricVehicle {
  type: Literal['electric_vehicle'] = 'electric_vehicle'
  charger: ElectricVehicleCharger
  weight_comfort: float
}

class VariableProductionAsset {
  max_acceptable_curtailment_rate: int | None = None
}

Asset <|-- NonFlexibleAsset
Asset <|-- FlexibleAsset
FlexibleAsset <|-- StorageDevice
FlexibleAsset <|-- FlexibleLoadAsset
FlexibleAsset <|-- DispatchableProductionAsset
NonFlexibleAsset <|-- NonFlexibleLoadAsset
NonFlexibleAsset <|-- VariableProductionAsset
IntEnum <|-- RiskAggressiveness
StrEnum <|-- Direction
Schedule <|-- AssetSchedule
StorageDevice <|-- BatteryStorage
FlexibleLoadAsset <|-- ElectricVehicleCharger
FlexibleLoadAsset <|-- HVAC
BatteryStorage <|-- ElectricVehicle
VariableProductionAsset <|-- PV
```

...

B.3 Algorithm 1-1: Centralised energy management system for Community Energy Management System

```
```mermaid
```

```
classDiagram
```

```
class EnergyManagementSystemCommunityOutput {
 asset_dispatches: list[AssetSchedule]
 grid_offtake: list[Schedule]
 grid_injection: list[Schedule]
}
```

```
class OptimizationInstance {
 instance_id: str
}
```

```
class CommunityMember {
 id: str
 meters: list[Meter]
}
```

```
class Household {
 community_members: list[CommunityMember]
 assets: list[PV | BatteryStorage | ElectricVehicle | HVAC | NonFlexibleLoadAsset |
FlexibleLoadAsset]
 risk_aggressiveness: RiskAggressiveness = 50
}
```

```
class Schedule {
 schedule: list[TimeValue]
}
```

```
class EnergyManagementSystemCommunityInput {
 time_units: TimeParameters
 households: list[Household]
 members: list[CommunityMember]
 objective: list[Objective]
 community_manager: EnergyCommunityManager
 schedules: list[AssetSchedule]
}
```

```
class EnergyCommunityManager {
 id: str
 contract: Contract
}
```

```
class TimeParameters {
 timestep_minutes: int
 horizon_hours: int
 scheduling_time: datetime
}
```

```
class AssetSchedule {
 asset_id: str
 direction: Direction
}
```

```
class Objective {
 name: ObjectiveName
 weight: float | None = 1
}
```

```
OptimizationInstance <|-- EnergyManagementSystemCommunityOutput
```

```
OptimizationInstance <|-- EnergyManagementSystemCommunityInput
```

```
...
```

## B.4 Algorithm 1-2: Centralised energy management system with P2P preferences

---

```
```mermaid
```

```
classDiagram
```

```
class OptimizationInstance {  
    instance_id: str  
}
```

```
class CommunityMember {  
    id: str  
    meters: list[Meter]  
}
```

```
class ProductDifferentiationParcel {  
    trading_partners: set[str]  
    trade_coefficient: list[float]  
    trade_characteristic: list[list[float]]  
}
```

```
class PeerSharingCoefficient {  
    id: str  
    sharing_coefficient: float  
    sharing_price: float  
}
```

```
class EnergyManagementSystemP2POutput {  
    peer_results: list[PeerResult]  
    electricity_prices: list[TimeValue]  
}
```

```
class EnergyManagementSystemP2PInput {  
    time_parameters: TimeParameters = timestep_minutes=15 horizon_hours=24  
    scheduling_time=datetime.datetime(2025, 12, 4, 16, 42, 46, 643878)  
    peers: list[Peer]  
}
```

```
class Peer {  
    assets: list[Asset]  
    cost_function: QuadraticCostFunction  
    product_differentiation_parcel: ProductDifferentiationParcel  
    trading_partners: set[str]  
    flexibility_rate: float  
    availability: SharedAvailability  
}
```

```
class TimeParameters {  
    timestep_minutes: int  
    horizon_hours: int  
    scheduling_time: datetime  
}
```

```
class PeerResult {  
    id: str  
    load_profile: list[TimeValue]  
    production_profile: list[TimeValue]
```

```
    sharing_distribution: list[PeerSharingCoefficient]
}
```

```
class Criterion {
    <<Enumeration>>
}
```

```
class SharedAvailability {
    min_availability: list[TimeValue]
    max_availability: list[TimeValue]
    direction: bool
}
```

```
class TimeValue {
    time: datetime
    value: float
}
```

```
class Asset {
    id: str
    meter: Meter | None = None
    p_min_kw: float | None = None
    p_max_kw: float | None = None
    context: dict = dict
}
```

```
class QuadraticCostFunction {
    a: float
    b: float
    c: float
}
```

CommunityMember <|-- Peer

OptimizationInstance <|-- EnergyManagementSystemP2PInput

OptimizationInstance <|-- EnergyManagementSystemP2POutput

...

B.5 Algorithm 1-3: Centralised energy management system for benefit allocation

```
```mermaid
classDiagram
 class OptimizationInstance {
 instance_id: str
 }
 class CommunityMember {
 id: str
 meters: list[Meter]
 }
 class BenefitAllocation {
 id: CommunityMember
 sharing_coefficient_result: list[TimeValue]
 }
 class Schedule {
 schedule: list[TimeValue]
 }
```

```
class EnergyManagementSystemCollectiveBenefitAllocationInput {
 asset: list[Asset]
 asset_dispatches: list[AssetSchedule]
 grid_offtake: list[Schedule]
 grid_injection: list[Schedule]
 members: list[CommunityMember]
 member_preferences: list[BehaviouralWeight]
 behavioural_constraint: BehaviouralConstraint
}

class BehaviouralWeight {
 id: CommunityMember
 sharing_coefficient: float
}

class BehaviouralConstraint {
 <<Enumeration>>
 HARD_CONSTRAINT: str = 'hard constraint'
 SOFT_CONSTRAINT: str = 'soft constraint'
 NO_CONSTRAINT: str = 'no constraint'
}

class EnergyManagementSystemCollectiveBenefitAllocationOutput {
 community_total_benefit: float
 benefit_allocations: list[BenefitAllocation]
}

class StrEnum {
 <<Enumeration>>
}
```

```
class TimeValue {
 time: datetime
 value: float
}

class AssetSchedule {
 asset_id: str
 direction: Direction
}

class Asset {
 id: str
 meter: Meter | None = None
 p_min_kw: float | None = None
 p_max_kw: float | None = None
 context: dict = dict
}

StrEnum <|-- BehaviouralConstraint
OptimizationInstance <|--
EnergyManagementSystemCollectiveBenefitAllocationInput
OptimizationInstance <|--
EnergyManagementSystemCollectiveBenefitAllocationOutput
...

```

## B.6 Algorithm 1-4 Centralised energy management system – Individual portfolio optimisation DTO

---

```
```mermaid
```

```
classDiagram
```

```
class OptimizationInstance {  
    instance_id: str  
}
```

```
class Household {  
    community_members: list[CommunityMember]  
    assets: list[PV | BatteryStorage | ElectricVehicle | HVAC | NonFlexibleLoadAsset |  
FlexibleLoadAsset]  
    risk_aggressiveness: RiskAggressiveness = 50  
}
```

```
class Schedule {  
    schedule: list[TimeValue]  
}
```

```
class EnergyManagementSystemIndividualPortfolioInput {  
    household: Household  
    community_contract: CommunityContract | None = None  
    objectives: list[Objective] | None = None  
    start_datetime: datetime | None = None  
    end_datetime: datetime | None = None  
}
```

```
class ExternalInteractions {  
    grid_offtake_volume: Schedule
```

```
    grid_injection_volume: Schedule
    community_offtake_volume: Schedule
    community_injection_volume: Schedule
}

class CommunityContract {
    supply_prices: Prices
    resale_prices: Prices
    max_supply_community: Schedule
    max_offtake_community: Schedule
}

class AssetSchedule {
    asset_id: str
    direction: Direction
}

class EnergyManagementSystemIndividualPortfolioOutput {
    dispatches: list[AssetSchedule]
    external_interactions: ExternalInteractions
}

class Prices {
    prices: list[TimeValue]
}

class Objective {
    name: ObjectiveName
    weight: float | None = 1
}
```

```
OptimizationInstance <|-- EnergyManagementSystemIndividualPortfolioInput
OptimizationInstance <|-- EnergyManagementSystemIndividualPortfolioOutput
```

...

B.7 Algorithm 2: Centralised market clearing

```
```mermaid
```

```
classDiagram
```

```
class OptimizationInstance {
 instance_id: str
}
```

```
class LinkedOrderRelations {
 <<Enumeration>>
 PARENT: str = 'parent'
 CHILD: str = 'child'
 EXTENDED_PARENT: str = 'extended-parent'
 EXTENDED_CHILD: str = 'extended-child'
 LOOP_BUY: str = 'loop-buy'
 LOOP_SELL: str = 'loop-sell'
 FLEXIBLE: str = 'flexible'
}
```

```
class FlexibleBlockOrder {
 block_start: datetime | None
 block_end: datetime | None
}
```

```
class ClearingPriceModel {
```

```
}
```

```
class StrEnum {
 <<Enumeration>>
}
```

```
class OrderCurve {
 order_id: str
 linked_order_id: str | None = None
 price_volume_curve: list[OrderStep]
}
```

```
class MarketClearingOutput {
 cleared_orders: list[ClearedOrder]
 clearing_price: list[TimeValue]
}
```

```
class OrderStep {
 volume_min: float
 volume_max: float
 price: float
}
```

```
class TimeStepOrder {
 timestep: datetime
}
```

```
class BlockOrder {
 timesteps: list[datetime]
 relation: LinkedOrderRelations
}
```

```
class MarketClearingInput {
 orders: list[Order]
 objective_ranks: list[ObjectiveRanking] | None = None
 clearing_price_model: ClearingPriceModel | None = None
}
```

```
class ObjectiveName {
 <<Enumeration>>
 SOCIAL_WELFARE: str = 'social_welfare_maximisation'
 CLEARED_VOLUME: str = 'cleared_volume_maximisation'
 COST_MINIMISATION: str = 'cost_minimisation'
 MAX_VOLUME_SQUARED_DEVIATION: str = 'max_volume_squared_deviation'
 GRID_SUPPLY: str = 'grid_supply_minimisation'
 RENEWABLE_SUPPLY: str = 'renewable_supply_maximisation'
}
```

```
class ObjectiveRanking {
 name: ObjectiveName
 rank: int
 slack: float = 1.0
 direction: bool = None
}
```

```
class ClearedOrder {
 member_id: str
 is_offer: bool
 cleared_timestep_orders: list[TimeStepClearedOrder]
}
```

```
class TimeValue {
```

```
time: datetime
value: float
}
```

```
class TimeStepClearedOrder {
 cleared_volume: float
 cleared_price: float
}
```

```
class Order {
 member_id: str
 is_offer: bool
 timestep_orders: list[TimeStepOrder]
}
```

```
StrEnum <|-- ObjectiveName
StrEnum <|-- LinkedOrderRelations
OrderCurve <|-- TimeStepOrder
OrderCurve <|-- BlockOrder
BlockOrder <|-- FlexibleBlockOrder
OptimizationInstance <|-- MarketClearingOutput
OptimizationInstance <|-- MarketClearingInput
TimeStepOrder <|-- TimeStepClearedOrder
```

...

## B.8 Algorithm 3: Pricing mechanism

---

```
```mermaid
```

```
classDiagram
```

```
class OptimizationInstance {  
    instance_id: str  
}
```

```
class PricingMechanismOutput {  
    internal_prices: list[TimeValue]  
    billing: list[BillingReprocessing]  
}
```

```
class PricingMechanismInput {  
    time: TimeParameters  
    pricing_rule: PricingRule  
    loads: list[TimeValue]  
    productions: list[TimeValue]  
    prices: RetailPrices  
}
```

```
class BillingReprocessing {  
    contract_id: str  
    payback_amount: float  
    supplier_id: str | None  
    notes: str | None  
}
```

```
class TimeParameters {
```

```
timestep_minutes: int
horizon_hours: int
scheduling_time: datetime
}
```

```
class PricingRule {
  <<Enumeration>>
  MID_MARKET: str = 'Mid-market rate'
  SUPPLY_DEMAND: str = 'Supply-demand ratio'
}
```

```
class StrEnum {
  <<Enumeration>>
}
```

```
class TimeValue {
  time: datetime
  value: float
}
```

```
class RetailPrices {
  injection_price: list[TimeValue]
  offtake_price: list[TimeValue]
}
```

```
StrEnum <|-- PricingRule
OptimizationInstance <|-- PricingMechanismInput
OptimizationInstance <|-- PricingMechanismOutput
```

...

B.9 Algorithm 4: Heuristic benefit allocation

```
```mermaid
```

```
classDiagram
```

```
class ContractShare {
 contract_id: str
 share: float
}
```

```
class CommunityMember {
 id: str
 meters: list[Meter]
}
```

```
class OptimizationInstance {
 instance_id: str
}
```

```
class Schedule {
 schedule: list[TimeValue]
}
```

```
class CommunityMemberGridSchedule {
 community_member: CommunityMember
 grid_offtake: list[Schedule]
 grid_injection: list[Schedule]
}
```

```
class HeuristicBenefitAllocationInput {
```

```
 sharing_keys: list[ContractShare]
 community_member_schedules: list[CommunityMemberGridSchedule]
 total_benefit: float
 }
```

```
class HeuristicBenefitAllocationOutput {
 member_benefits: list[CommunityMemberBenefit]
}
```

```
class CommunityMemberBenefit {
 community_member: CommunityMember
 interval_start: datetime
 interval_end: datetime
 benefits_redistribution: float
}
```

```
OptimizationInstance <|-- HeuristicBenefitAllocationInput
OptimizationInstance <|-- HeuristicBenefitAllocationOutput
```

...