



ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

NEUMANN JÁNOS
INFORMATIKAI KAR



Development and comparison of optimisation algorithms for energy management

**OE-NIK
2024**

Student
name

Registration number:

Tolnai Gergő

T009332/FI12904/N

ABSTRACT

Local Energy Communities (LECs) are groups of citizens, small enterprises or local authorities that have access to locally produced energy, typically from renewable resources such as photovoltaic panels or wind turbines. These communities face various optimisation challenges related to energy management, maintenance, energy logistics, and investment decisions. One of the primary objectives of LECs is minimising electricity costs. This goal can be achieved through effective energy management, which optimises the use of local energy resources and battery energy storage systems (BESS). The energy management system must determine the optimal schedules of battery usage and external power purchases based on consumption, renewable generation and price forecasts to minimise overall energy costs. Various algorithms have been proposed to support these decision-making processes.

In this paper, we implemented three distinct algorithms from the literature that address this optimisation problem. We evaluated the cost performance of each algorithm and its variations using a test environment based on publicly available historical datasets from Hungary, including solar energy production time series data, consumption profiles, and hourly day-ahead market prices. The algorithms were compared based on their economic optimization capability, computational complexity and input data requirements. After assessing the performance of the algorithms, we made recommendations for the most suitable solutions for different scenarios and use cases.

CONTENTS

1	INTRODUCTION	1
2	LITERATURE RESEARCH	3
2.1	Main Components of Local Energy Communities	3
2.2	Business models.	4
2.3	Pricing models	6
2.4	Energy Cost optimisation algorithms.	7
2.4.1	Greedy strategy	7
2.4.2	Rule-based algorithm.	7
2.4.3	Exact and Heuristic optimisation algorithms.	10
2.5	Notes on collective or individual installation.	13
3	METHODOLOGY	14
3.1	Objective and Constraints	14
3.2	The working environment	15
3.3	Data sources and processing	17
4	IMPLEMENTATION OF THE ALGORITHMS	20
4.1	PV only	20
4.2	Greedy algorithm	20
4.3	Rule-based algorithm.	21
4.4	Linear Programming solver algorithm	23
4.5	Operation	24
4.6	Testing	24
5	RESULTS.	29
5.1	Rule based hyper parameter tuning	29
5.2	Comparison of all strategies	30
5.3	Performance comparison using varying time intervals	38
5.4	Using Linear Programming for Net Revenue Optimisation	39
5.5	Comparison of Greedy and Rule-based algorithm.	41

5.6 Overall comparison	44
6 FUTURE DEVELOPMENT	47
7 ACRONYMS	50
8 BIBLIOGRAPHY	51
9 LIST OF FIGURES	54

1 INTRODUCTION

Local energy communities are groups whose main goals are to cooperate on energy production and consumption in order to achieve more cost-efficient energy usage or minimise dependence on outside power sources. They use renewable sources such as photovoltaic and wind power to produce their own energy. Alongside these initial goals, developing and operating an efficient local energy community presents many opportunities for optimisation. Various optimisation strategies and algorithms have been introduced to encompass not only the initial design and component selection but also the ongoing operations of the system, such as how to manage locally produced energy in the most cost-effective or sustainable manner. [1] places the different optimisation problems into four groups: Environmental, Social, Technical, and Economic.

The Social aspect primarily refers to the users' comfort or discomfort level with the energy supply, as Demand Side Management (DSM) implementations can alter household consumption patterns to reduce costs. This may cause dissatisfaction, as user preferences might not be met. These preferences can include the maximum and minimum levels of energy demand during specific periods, with discomfort measured by the deviation from the preferred maximum [2]. Environmental strategies focus on minimising the community's environmental footprint. Studies on this topic often include objectives like reducing CO_2 emissions, which typically correlates with reducing dependence on the power grid. These objectives are often part of a multi-objective optimisation that also targets energy costs [3], [4].

The technical category introduces metrics such as self-consumption rate and self-sufficiency rate. The self-consumption rate measures how efficiently a system uses its locally produced energy, indicating how much energy is self-consumed versus wasted or exported to the grid. Self-sufficiency measures a community's independence from external energy sources and is the ratio of locally produced energy to the total energy consumed [5].

According to [1], the most commonly addressed optimisation problems fall within the Economic group. Among these, minimising overall energy costs is the most prevalent objective for energy communities. Therefore, this study focuses on reducing energy costs and exploring the algorithms designed to achieve this.

This study is part of an ongoing project at Óbuda University. My task was to develop a test environment, research different cost-optimisation algorithms from the literature, and then compare their performance. The test environment developed mirrors our educational institution, incorporating photovoltaic production, an aggregated consumption profile, a central battery energy storage system (BESS) for energy reservation, and day-ahead market energy pricing for external purchases. The production, consumption, and pricing data were collected from publicly available time-series datasets.

The production profile was selected from a photovoltaic unit closest to our institution, while the consumption profile was chosen to reflect an educational institution's consumption pattern. Hungarian Power Exchange's day-ahead market (HUPX DAM) pricing was used as a legitimate

source for energy market prices. The production profile was re-scaled to create various test cases with different production rates. Following the test environment's development, I have selected cost-optimisation algorithms that were applicable. Sub-versions of these algorithms were also implemented, and their cost-reduction performance was tested using the collected test data.

Finally, the algorithms were compared based on the experimental results, their complexity, and their data requirements. In the test, I measured the overall energy costs and the amounts of electricity imported and exported to the external grid. Net revenue from electricity sales was also considered to explore a potential use case where the community could profit from the energy market. The relative complexity of each algorithm was evaluated to determine ease of implementation. Additional data requirements, such as forecast data, were also assessed to determine which algorithms introduce more challenges or development needs. The goal of this study is to identify the most suitable algorithms for different use cases and to highlight which algorithms or their sub-versions may be less effective based on the comparison criteria.

2 LITERATURE RESEARCH

First and foremost this research must include the different components of that a Local Energy Community (LEC) might have. It is important to know what components can an algorithm utilise to achieve its goal. Secondly the different business models will be explored that describes the business relation between the community members. The chosen model should be the closest to a model that an educational institution might have. Additionally it should match the optimisation goal that aims to reduce the **collective** energy cost of a community.

2.1 Main Components of Local Energy Communities

Local energy communities (LEC) are communities at place. They are usually groups of people in a defined area, such as a neighborhood that is connected together with a distribution grid. In a local energy community, members can potentially achieve a certain level of energy independence by collectively producing some or all of the power they consume through the renewable energy sources. The primary components that a Local Energy Community may include are energy generation units, a Battery Energy Storage System (BESS), a local grid connecting all elements, households (i.e., consumers), and an Energy Management System (EMS), which is primarily responsible for load balancing and energy distribution, as illustrated in figure 2.1.

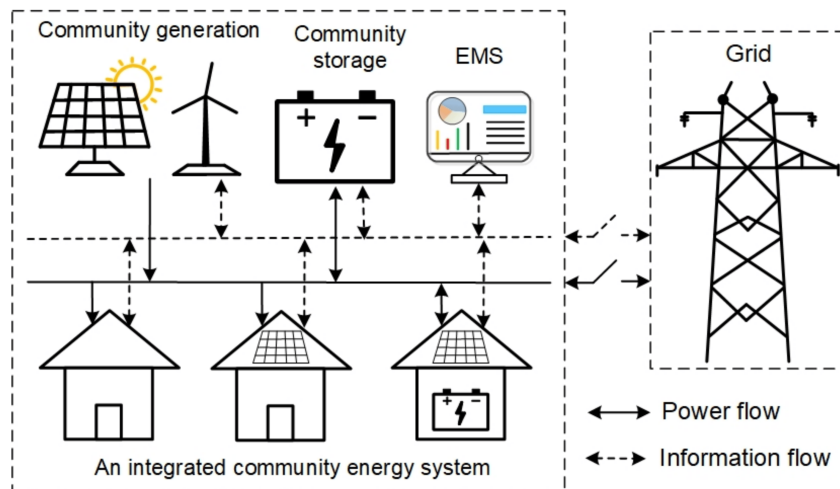


Figure 2.1: LEC system with central and individual battery and energy generation [6]

Energy generation: For energy generation there are many other sources of energy that can be used, however the most widespread solution is the use of Photovoltaic panels (PV). For this reason I will only include solar panel production in my work.

BESS: Battery Energy Storage System or BESS is useful for different reasons, whenever the community has production surplus during off-peaks, it can be stored in batteries that can be accessed during peak times when energy production might not have enough output to satisfy

demand. There is also a practice when they purposefully charge their battery from the main grid when it is the cheapest [7].

Production and storage can be solved by central or individual installations as well. As it can be seen on figure 2.1 there are cases when the two solutions are combined. It mainly depends on the community members preferences and goals. The chosen business model can also be a decisive factor, as I will discuss it in later sections.

Local grid: The local grid plays a crucial role in distributing energy among the members of the LEC. It serves as the connection between the PV system, the BESS, and the households. Additionally, it is essential for the local grid to provide connectivity with the external network to ensure a reliable energy supply when local production is insufficient to meet electricity demand.

EMS: EMS is the brain of the system, it monitors, controls, and optimises the operation of all the components in the local energy community to ensure efficient energy use. It collects the aforementioned data that is fed to the optimisation algorithm used for real time decision making. Different optimisation algorithms and objective functions are used here. Finally all communities must have a connection to the outside grid for cases when they cannot cover their needs by self-generation and need to purchase electricity from outside providers.

2.2 Business models

When a community is founded, it can use different business models (BM) to create or fund their system. For instance, there is difference from model to model regarding generation unit ownership. Furthermore, there are models that mainly support the members on an individual level, and others that support the community as a whole. It is safe to say that the underlying business model determines how we calculate the overall cost of operation. For example, if we want to optimise on energy cost, there is difference in the cost function between communities who include a Local Energy Market and individual production, allowing users to trade energy within the system. And systems that have centralised generation units and batteries which are collectively owned by the whole community.

From the models referenced in [8] and [1] here are some of the most important:

- Prosumerism
- Local Energy Markets (LEM)
- Collective generation
- Aggregator

Prosumerism: Prosumerism is a business model where the members have their own generation units such as PV, as well as storage systems like BESS. However additional shared systems can also be present. They are typically a local community in close proximity to each other. They

can play the role of investors, customers and decision-makers. Their main goal is to cover their need via self-consumption of renewable energy and share their excess energy within the bounds of the community usually through a LEM. It is a good practise to aggregate their demand and surpluses in order to increase their bargaining power when negotiating with retailers. Since individually, as separated small consumers they do not have that much impact on the market as in the case of joining into a single large consumer unit, buying electricity in bulk purchases, usually in a lower rate. Surplus energy can be sold and reinvested into the community or distributed between members to reinvest in their individual systems. [1]

Local Energy Markets Local Energy Markets are usually part of prosumer communities, in order to maximize self-consumption via local energy trades, so that they minimise the dependence on external sources. They can work collaboratively to maximise self-sufficiency, by covering each other's needs through trades on the local market or they can establish a competitive market, where they are pursuing the most individual cost-efficient approach of energy usage and trading including external sources. The second option might not result in better self-sufficiency because in this case buyers might consider buying energy from the grid when it's exceptionally cheaper than usual, charging their BESS for later times, when it might be needed. As for example, supply side rule-based algorithms also include a lower price threshold to charge the BESS when it is considered 'cheap' enough [7], [4]. In LEM members share their produced energy in a peer-to-peer manner. Prices can be negotiated between the users. Prosumers can choose who they sell their energy to, and consumers can choose from whom they want to buy.

Collective generation: These communities have shared energy generation and storage system. These units can be installed on roof-tops of multi-tenancy building or nearby consumers. This BM can be applied for communities of place [8], [1]. The main goal of this type of constitution is to cover all the demands of the community members via self-generated renewable energy. The users can benefit from reduced electricity bills and also the revenue from the sold surpluses, which are shared among them. These BMs are getting popular in the EU. For example the projects Bostadsrättsföreningen Lyckansberg (Sweden) and Solar roof (Bulgaria) focuses on energy generation and sharing, while Solbyn Association (Sweden) produces heat/hot water for its members [8].

Aggregator: Users in this model are grouped by aggregators, whose role is to act as an intermediary between the community members and flexibility markets, allowing even smaller communities to access them. The main goal behind this BM is to allow communities to engage into collective DSM strategies to provide demand flexibility to grid operators [8].

Flexibility market is a platform where participants can sell their "flexibility". That basically means two things: the ability to adjust one's consumption pattern to lower their demand during peak times or they can sell their unused capacities of renewable energy whenever they generate surplus. These markets are made for larger entities such as industrial and commercial players

as they have significant power consumption and demand meaning a larger amount of flexibility, moreover they have the sufficient capital needed to invest in large generation plants and necessary tools that make consumption and production measurements which are needed in order to join. For smaller energy producers like family households these are impossible requirements to meet. Aggregators are therefore the solution, they can group the individual households into a pool, controlling all of their capacities as a whole. This way they can achieve volumes that enable them to participate in these markets.

Aggregators may provide the necessary sensors, smart meters and even HEMS systems to control the household. Contracts can be signed between the end-user and the aggregator such as dispatchable DSM or non-dispatchable DSM programs. In the first users accept external control over their appliances during peak hours via direct load control. The latter program allows users to receive dynamic pricing data in real time, allowing them to adjust their consumption profile accordingly [8]. It can open new optimisation opportunities for end-users as many cost-optimisation algorithms are based on real time pricing, such as rule-based algorithms [7], [9].

An educational institution such as our university can be considered as an energy community with a single large entity. Energy cost optimisation can be separated into two parts: Supply side and demand side cost optimisation. While demand side management changes the consumption pattern, supply side management focuses on the available options for covering demand that consists on local reserves in BESS and outside energy markets. Communities can consider focusing on only one or both sides of optimisation. This paper only considers supply side optimisation algorithms, therefore the closest model that could be used is **Collective generation**. We aggregate all the electricity needs and energy production of the institution to determine the overall demand.

2.3 Pricing models

There are different ways of measuring the cost of the consumed energy. The main pricing models are Fixed Pricing, Time of Use (ToU) and Real Time Pricing (RTP).

Fixed pricing speaks for itself. Consumers pay a predetermined, flat rate of used up electricity from the grid, the price will not change during the day. Time of use however, divides the day into different time periods assigning these periods with different price tariffs. As for an example the cost optimisation in [9] is based on ToU. The model has four time periods: peak (from 6 p. m. to 10 p.m), shoulder (1 a.m. and from 9 a.m. to 5 p.m. and from 11 p.m. to 12 p.m.) off-peak (2 a.m. and from 7 a.m. to 8. a.m.) and super off-peak (from 3 a.m. to 6 a.m.) The price tariff is the highest during the peak period and the lowest on super off-peaks. Real time pricing (RTP) is a dynamic pricing model that follows the fluctuation of supply and demand in real time. RTP requires investments in smart technologies such as HEMS.

The comparison of the three models in [4] suggests that cost optimisation models benefit the most from using Real Time Pricing. It allows users to buy electricity at times when it is significantly lower for reuse during more expensive periods. This is the aspect that optimisation

algorithms tend to leverage with the usage of BESS as I have mentioned on page 4. Therefore the test environment should include a similar solution for determining energy price.

2.4 Energy Cost optimisation algorithms

In the following, I will examine energy cost-optimisation algorithms as part of a literature review, progressing from simpler to more complex ones.

2.4.1 Greedy strategy

In [10] two algorithms are proposed to manage the community's aggregated energy need. The first was the so called Greedy algorithm. It is the simplest implementation of energy management. The strategy is the following, it calculates the difference between the overall production and consumption in each time interval. If there is an energy surplus, meaning that the production is greater than the demand then the algorithm loads as much energy from the surplus as possible into the battery. The remaining energy, if any, is then exported to the grid. Otherwise, if there is an energy deficit first it covers the energy demand from the battery, and if it is not sufficient, then electricity is imported from the grid. The Greedy naming comes from the fact that the method makes decisions based on what is the best solution in the given moment, unlike the following algorithms that also take future data to consideration when making decisions on energy management.

2.4.2 Rule-based algorithm

The supply side optimisation strategies usually rely on real-time electricity grid pricing. They control when to buy electricity needed by the community to reduce costs. The studies I have examined, [4] and [7] both implement a rule-based strategy, that encourages purchasing electricity when it is cheaper, reserving it for peak periods. Before examining the optimisation proposed in the two works mentioned, let us assess the basic differences. Both approaches divide the given time period into time intervals making decisions whether to buy, sell or charge, discharge the battery in each interval, however [7] has an hourly granularity, while [4] divides the day into 10 minute time slots.

[7] optimises self-consumption and energy cost for a whole energy community while [4] only focuses on a single household. Therefore the first study had to account for allocating the collectively generated energy between the loads. There are two approaches: dividing the collective power with the use of static or variable coefficients. We order a fraction smaller than 1 to each load in a way that they add up to 1:

$$\sum_{j=1}^J c_j = 1 \quad c \in C \quad (2.1)$$

$$c_j = 0.1\mathbb{N}^* \quad \forall c \in C, j \in J \quad (2.2)$$

The Equations are from [7]. C is the set of all coefficients for each load, J is the number of loads. 2.2 constraint the use of coefficients to only use positive numbers with the multiples of 0.1. These fractions indicate the proportion of the generated energy that is allocated to the given load. With the use of static coefficients we define a fix proportion for each load which will remain constant for the whole period. With this solution in order to charge the BESS, we need to assign a coefficient that load as well. Variable coefficients are different in a way that they can change over time within the period. The study determined these coefficients by going through all the possible combinations of coefficients in order to find a solution that generates the most significant bill savings.

Testing both cases in [7] they found out that variable coefficients gives more flexibility to the LEC making it less dependent on the grid improving the communities footprint while also reducing energy costs. Pairing this solution with prohibiting the charging of BESS from the grid can result in a significantly more environmentally friendly consumption. However the billing savings achieved by variable coefficients are nullified by the fact that it also shortens battery life as this approach uses BESS more intensively. It can result in more costs on the long run. Whereas [4] prohibits selling surpluses to the grid, [7] does not have such constraints and benefits from the revenues generated by selling. Therefore there will be a small difference between the two rule-based approaches implemented by this two studies, in any other respects the two algorithms are the same.

For each time interval they first determine the demand for each load in the community comparing it to the energy allocated from PV production, or in the case of HEMS, the comparison is made against the energy need for all appliances:

$$\sum_a^A p_t^a < p_t^{PV} \quad \forall t \in T \quad (2.3)$$

The inequality presented is from [4]. In this case a stands for a home appliance from A , the set of all appliances in the smart home. p_t^a means the amount of energy needed for home appliance a in the time interval t . If the inequality is true, meaning that there is more generation allocated than the demand, that means energy surplus, otherwise it can result in energy deficit. Therefore, the algorithm branches into two options to address both cases.

On the surplus branch they first examine the BESS's state of charge (SoC). If it is full, then they sell all the surplus to the grid. Otherwise they charge as much surplus energy as they can into the BESS, the remaining surplus is then sold to the grid except in the case of [4]. If after the surplus charge the BESS remains not completely full, the algorithm checks the current energy price comparing it to a parameter called the lower threshold or LT [7]. This parameter's purpose is to determine whether the current energy price is cheap enough in order to charge the BESS from the grid for later use. Assigning of this parameter can be different for all algorithms. In [7] it is the 25th percentile of the last 15 days' electricity prices.

On the deficit branch they first charge all the allocated energy into the given load, then

calculating the difference from the subtraction of the demand and the allocated energy:

$$\Delta p_2 = \sum_a^A p_t^a - p_t^{PV} \quad (2.4)$$

The equation presented is from [4]. We then compare the energy price to a parameter called Upper Threshold or UT. If the electricity price is lower than the given threshold the system charges electricity from the grid, otherwise if it is more expensive, then it discharges from the BESS in order to cover the remaining demands. If the energy stored was not sufficient, then again, the remaining amount is charged from the grid. In [7] UT is determined by an upper percentile of the day's energy prices. The proper percentile for each season can be different. This is for a reason. It is better to have a stricter percentile during winter than summer because of the lower solar resource. They limited the possible percentiles to be the multiples of 25 to save time on calculating the optimal solution. In [7] the proper percentile for each season are determined the same way as for the load allocation coefficients. Constraints have been introduced to the mathematical model. From [7] namely the sum of the energy sources must match the sum of sinks:

$$(p_t^{PV} + p_t^{B,D} + p_t^{PUR})\Delta t = (p_t^D + p_t^{B,C} + p_t^{SELL})\Delta t \quad (2.5)$$

Where p_t^{PV} , $p_t^{B,D}$ and p_t^{PUR} are the power produced from PV, power discharged from battery and power purchased from the grid respectfully, while p_t^D , $p_t^{B,C}$ and p_t^{SELL} stand for the power demanded, the power charged to the battery and the power sold to the grid. Δt is the length of the time slot. Furthermore the self-consumed energy at the moment cannot be any other than the lower of the allocated energy or the demanded energy:

$$p_{j,t}^{PV} = \min(p_{j,t}^A, p_{j,t}^D) \quad \forall t \in T, j \in J \quad (2.6)$$

Where $p_{j,t}^A$ stands for the allocated energy for the load j at the moment t .

BESS constraint are also introduced addressing their capacity and charging/discharging capabilities:

$$p_t^{B,D} p_t^{B,C} = 0 \quad \forall t \in T \quad (2.7)$$

$$0 \leq p_t^{B,C} \leq p_{MAX}^{B,C} \quad \forall t \in T \quad (2.8)$$

$$0 \leq p_t^{B,D} \leq p_{MAX}^{B,D} \quad \forall t \in T \quad (2.9)$$

$$0 \leq E_t^B \leq E_{CAP}^B \quad \forall t \in T \quad (2.10)$$

2.7 prevents the system from simultaneously charging and discharging the battery. While the equations 2.8, 2.9 set an upper limit of charging and discharging in any moment t with the maximum charging and discharging rates given. 2.10 sets the maximum capacity of the BESS. All the previous equations mentioned are from [7].

2.4.3 Exact and Heuristic optimisation algorithms

In [1] energy cost-optimisation algorithms have been classified into **exact** and **heuristic** categories with the following examples:

- Exact:
 - Linear Programming (LP)
 - Mixed-Integer Linear Programming (MILP)
 - Alternating Direction Method of Multipliers (ADMM)
- Heuristic:
 - Particle Swarm Optimisation (PSO)
 - Genetic Algorithm (GA)

Exact algorithms Exact algorithms, as they name suggests can provide the exact global optimal solution for a given problem if the required criteria is satisfied. Namely if the problem can be modelled in a linear mathematical formula, then LP can be used. However for non-linear problems, if the objective function is convex then ADMM can be implemented.

Linear Programming and Mixed-Integer Linear Programming mathematical modeling techniques in which a linear function is maximized or minimized when subjected to various constraints [11]. The criterion is that the variables used in both the objective function and the constraints should be raised to the power of 1:

$$f = c_1x_1 + c_2x_2 + \dots + c_nx_n \quad (2.11)$$

The function is subject to a set of constraints expressed in inequalities to prevent surreal solutions for the given problem:

$$c_{11}x_1 + \dots + c_{1n}x_n \leq b_1 \quad (2.12)$$

$$\vdots \quad (2.13)$$

$$c_{k1}x_1 + \dots + c_{kn}x_n \leq b_k \quad \text{with: } \forall x_i \geq 0 \quad (2.14)$$

[11]

As an example in [10] the second algorithm proposed was linear programming. It is used to determine the optimal solution with forecast data in a limited time window. The main difference between MILP and LP is that linear programming only allows continuous decision variables while mixed-integer programming allows both discrete and continuous values. The discrete values are usually in the set of 0, 1 which is used to for logical yes/no decisions, such as deciding whether to turn on or of a device in the given time period. Such example has been seen in [4].

ADMM is an algorithm for finding the optimum for convex functions. It decomposes the objective function into smaller, more manageable subproblems that can be optimized separately.

Then the problems' solutions are coordinated to find the original problem's optimal solution [12, p.3]. Assume that the objective function is separable to convex functions $f(x)$ and $g(z)$. Our goal is to minimize their sum:

$$\min f(x) + g(z) \quad (2.15)$$

Where the problem is subjected to the constraint:

$$Ax + Bz = c \quad (2.16)$$

Where:

$$\begin{aligned} x &\in R^n \\ z &\in R^m \\ A &\in R^{p \times n} \\ B &\in R^{p \times m} \end{aligned}$$

Meaning that x and z are vectors. The problem is then formulated as the following:

$$\min L_\rho(x, z, y) = f(x) + g(z) + y^T(Ax + Bz - c) + \frac{\rho}{2}\|Ax + Bz - c\|^2 \quad (2.17)$$

It is referred to as the **Augmented Lagrangian form**. Dual variable y is introduced to enforce the constraint on the original problem. $\|Ax + Bz - c\|^2$ is the squared norm of the vector evaluated by the expression, and ρ is called the penalty parameter.

This problem then can be decomposed into separate subproblems:

$$x_{k+1} := \underset{x}{\operatorname{argmin}} L_\rho(x, z_k, y_k) \quad (2.18)$$

$$z_{k+1} := \underset{z}{\operatorname{argmin}} L_\rho(x_{k+1}, z, y_k) \quad (2.19)$$

$$y_{k+1} := y_k + \rho(Ax_{k+1} + Bz_{k+1} - c) \quad (2.20)$$

The subproblems are solved iteratively with each subproblem evaluating only the primal variables x and z and the dual variable y respectfully. The term *Alternating Directions* comes from the fact that the variables x and z are evaluated in an alternating fashion. The algorithm approaches to the optimal solution as k approaches to infinity. With the use of suboptimal bounds we can set a tolerance level around the optimal solution, therefore the algorithm finishes within a finite number of iterations [12, p.13].

As an example in [13] the energy community consists of prosumers and a shared energy storage system. The study empathises a non-cooperative game model as all prosumers want to minimise their own cost however they all share the same storage system. The optimisation problem is to find the equilibrium between all the users' storage allocation and consumption strategies, minimising all of their objective functions as much as possible. To achieve this, first ADMM is used to determine each user's best response to the examined time period.

Heuristic algorithms Heuristic algorithms search for good solutions by exploring the solution space iteratively searching for better solutions in each iteration, hence the name. This behavior causes the algorithm to close in to a local optimum if any, meaning that a solution provided by such algorithms is not guaranteed to be the global optimum, none the less they can be effective. The two algorithms mentioned in this category are the Particle Swarm Optimisation (PSO) and Genetic algorithm (GA).

PSO algorithm PSO is a swarm algorithm that consist of a searchspace with n dimensions with multiple agents within it. The angents' goal is to find the optimal zone of the space. Each agents' position in the space represents a solution to the given problem. The agents iteratively converge towards the optimum by ajdusting their position and velocity in each iteration. For each agent adjustments are influenced by the current global best solution they have found and their individual best solution.

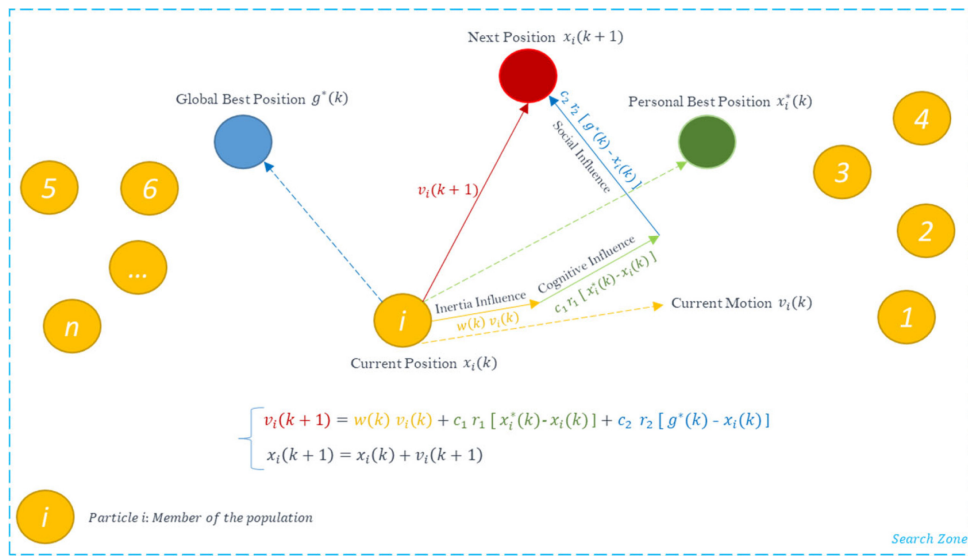


Figure 2.2: Example for PSO searchspace [4]

[1] used PSO to solve a multi-objective problem, where the households flexible loads were scheduled in each time interval. The different coordinates corresponds to the different home appliances in each timeslots. The literature states that this algorithm can avoid being stuck around a local optimum by the nature of using global best solutions in the calculation of the next iteration.

Genetic Algorithms (GA) are optimisation algorithms that have been introduced as part of evolutionary computation that mimics the biological processes such as reproduction, mutation and natural selection. This way the optimal solution is approximated with the agents' evolution through several generations. All GAs must include a fitness function, a population of agents, a method for selection, crossover and random mutations. [14].

In this project's case the optimisation objective and the constraints can be defined with a linear equation and inequalities. [10] similarly to my work, had collective consumption and

production data, and used linear programming to find the optimal energy usage in predefined time intervals that requires production, consumption and price forecasting. Therefore the third and last type of algorithm chosen for the comparison is Linear programming.

2.5 Notes on collective or individual installation

During my research I have reviewed studies that compared the cost-efficiency and self-consumption rate of collective BESS PV generation system with individual household installations. In [15] the authors implemented a model to simulate energy demand of communities. Furthermore they have developed a method to find the best design that maximises self-consumption and minimises costs. They evaluated an LEC comparing their results with individual PV installations for each household. Based on the results, they have concluded that a collective system is more cost-efficient. Similarly with BESS, in [7] the individual battery installation for each household has been compared with collective BESS shared by the whole community.

The results suggest that the collective installation is more effective regarding self-consumption. The reason for this is that even if some users' BESS gets fully charged, the remaining surplus is sold to the grid while other users' capacities are still not full. Unlike in the collective case, where surplus is sold only after the communities shared capacity is filled. These conclusions justify the use of aggregated demand and production data as well as a central BESS deployment in a cooperative setting such as the test environment that I have developed.

3 METHODOLOGY

The following algorithms have been chosen from the literature: Greedy, Rule-based and Linear Programming. These algorithms require collective consumption, production and price data in time series, as well as a BESS. With the aforementioned time series data each algorithm makes decisions in each time interval based on the state of the self-produced supply, demand, outside energy prices and battery state. The decisions can be represented by the following variables: $c_{in,t}$, $c_{out,t}$, ch_t and dch_t . Where $c_{in,t}$ and $c_{out,t}$ are the amount of imported or exported energy in time period t . Similarly, ch_t and dch_t are the amount of energy charged into or discharged from the battery at the given time period t . Each algorithm has its own method of decision making, however they have to respect the same constraint and their performance will be measured with the same objective function.

3.1 Objective and Constraints

The goal is the minimisation of the **overall energy cost**. So the objective function of the problem that has to be minimised by all of the algorithms is the following:

$$\text{Minimise } \sum_t^T c_{in,t} * p_t \quad (3.1)$$

Where T is the set of all the time intervals within the examined time period. $c_{in,t}$ is the amount of imported (purchased) energy during the time interval t . p_t is the unit price of electricity at time t .

Alongside the energy price, **Total revenue**, **Net revenue**, **Total export and import** is also measured. The total revenue is calculated from the selling price of the exported energy to the grid while net revenue is the difference of the total revenue and the total energy cost. Total export and import are self explanatory. They refer to the total amount of energy exported to and imported from the grid.

The test environment must include a BESS as mentioned. The parameters chosen for the battery system can be seen below:

- $SOC_{max} = 50 \text{ kWh}$
- $SOC_{min} = 0 \text{ kWh}$
- $CH_{max} = 12.5 \text{ kWh/h}$
- $DCH_{max} = 12.5 \text{ kWh/h}$
- $SOC_{start} = 0 \text{ kWh}$

SOC_{max} and SOC_{min} are the maximum and minimum allowed state of charge of the battery system measured in kWh. CH_{max} and DCH_{max} stands for the maximum charge and discharge rate in kWh per hour, respectfully. They define the maximum amount of energy that can be charged or discharged from the BESS in an hour. SOC_{start} is the starting state of charge at

the beginning of the measured period. In this experiment, all the algorithms will use a BESS with the same parameters. The maximum allowed capacity will be **50 kWh** and the minimum allowed state of charge will be **zero**. The charge and discharge rate will be equally **50 kWh per hour**, however we will use 15 minute intervals for each algorithm, that means that CH_{max} and DCH_{max} for a time interval of 15 minutes is equal to **12.5 kWh per hour**. The starting energy state (SOC_{start}) will always be the minimum allowed.

It is important to note that the allowed minimum should not be zero, nor should the maximum allowed state be below the physical capacity of the BESS. According to research [16], the recommended operational state of charge for the batteries is between 20% and 80%, aimed at preserving battery life. However, for the purpose of simplification, the effective minimum and maximum are set to 0 and 50 kWh, respectively.

Next the system's constraints have been defined relying on the research explained in 2.4.2:

$$ch_t \leq CH_{max} \quad \forall t \in T \quad (3.2)$$

$$dch_t \leq DCH_{max} \quad \forall t \in T \quad (3.3)$$

$$ch_t * dch_t = 0 \quad \forall t \in T \quad (3.4)$$

$$SOC_{min} \leq soc_t \leq SOC_{max} \quad \forall t \in T \quad (3.5)$$

$$soc_t = soc_{t-1} + ch_t - dch_t \quad \forall t \in T \setminus \{1\} \quad (3.6)$$

$$soc_1 = SOC_{start} \quad (3.7)$$

$$gen_t + dch_t + c_{in,t} = con_t + ch_t + c_{out,t} \quad \forall t \in T \quad (3.8)$$

$$c_{in,t} * c_{out,t} = 0 \quad \forall t \in T \quad (3.9)$$

The constraints from 3.2 to 3.4 are the constraints representing the charge and discharge limitations of BESS. 3.5 is BESS's capacity limitations while 3.6 links the battery's state of charge between the time intervals based on how much energy has been added or used. 3.7 sets the initial state of charge. Finally 3.8 is the balance rule between the sources and sinks, similarly to the constraint 2.5. gen_t and con_t are the energy produced and consumed at time interval t . 3.9 similarly to 3.4 limits the algorithms to either import or export at a given time period.

The experiment was made with some restrictions. This study only focuses on the electricity cost. Additional costs such as electricity contracts, BESS and PV maintenance costs, and infrastructure investment costs are out of the scope of this research.

3.2 The working environment

The project's modules and their connections can be seen on 3.1. I used **Jupyter notebook** for the development of this project for different reasons. First of all, it supports **Python**, which is a scripting language with libraries for data manipulation, processing and visualization. Therefore all scripts were coded in Python. The other reason is that Jupyter is capable of separating the code into segments with different purposes which are called cells. Depending on the content, cells can be run independently from each other. Additionally, the markdown language allows

me to take notes and labels between the cells, which can be used for code descriptions.

PySpark is used for data ingestion and processing. The necessary data is imported as raw csv files. These files are converted into Spark dataframes, on which the necessary consolidation transformations were made followed by joining the separate sources into a singular dataframe for the algorithms to handle. Spark is a reliable framework, suitable for batch processing that is also a useful mediator between different data sources that can be any kind of files or databases. I have chosen this tool for these particular reasons. **Cassandra** is one of these datasources that can be accessed through Spark. Therefore I have included this datastore as a storage module for the processed data sources for later use, therefore there is no need to rerun all the transformations every session. Every conventional database would work for this purpose, however Cassandra's partitioning allows for more efficient filtering on time series data, therefore it is a good choice for this project.

The algorithms were also programmed in Python, they use pandas dataframe as input. Pandas format allows iteration over its rows, which is necessary for the calculations as each time interval is dependent on the previous time interval's results. Pyomo is used for one of the algorithms that uses a linear programming solver. Pyomo is a modelling tool for linear programming optimisation problems that communicates with solvers. I have found two open source solvers that are available for free use, namely **glpk** and **cbc**. The latter is provided by the **COIN-OR** foundation [17]. It is a mixed-integer solver with a built-in linear programming solver called **clp**. The latter is chosen because of the mixed-integer capability, as constraints 3.4 and 3.9 are cannot expressed like this in Pyomo for LP solvers, instead I used support binary variables for enabling and disabling charging, discharging and exporting, importing. Following the calculations and the evaluations, the results and the different comparison visualisations are created with the **Matplotlib** and **Seaborn** libraries.

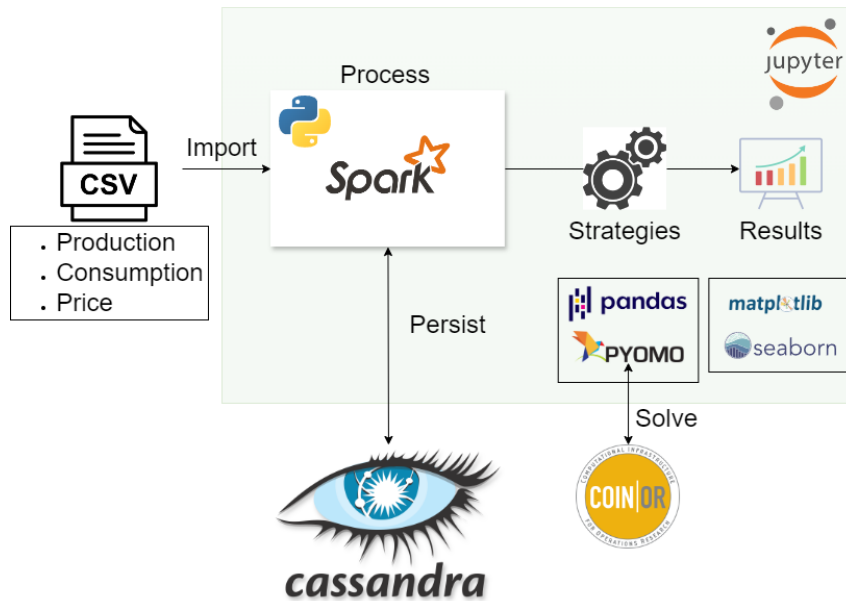


Figure 3.1: Working environment

The test environment was developed on a Virtual Machine with Ubuntu operating system using VMWare in order to isolate the workspace, and the ability to share the project with my associates. The system specifications are listed below in table 3.1

Specifications	
Host CPU	AMD Ryzen 7 5700U
Standard CPU frequency	1.8 GHz
Physical/logical cores	8/16
Host RAM	16 GB
Storage (HDD/SSD)	SSD 512 GB
Host Operating System	Windows 10
Virtual Machine Software	VMWare Player 17
VM Operating System	Ubuntu 22.04.3 LTS
VM RAM	4 GB
VM Cores	2

Table 3.1: System specifications

3.3 Data sources and processing

As stated before the algorithms require three crucial data sources: production, consumption and energy price. Additionally, the third algorithm, LP would also need additional forecasting of these data. However as forecasting is out of the scope of this project, it uses perfect foresight, or in other words, it is provided with existing data recorded in the past. In fact all the algorithm will iterate over the same dataset in order to compare their results on the same basis.

The test data is collected from three different sources between 2022 to 2023. The production's source is from Solar Edge public monitoring page. I have chosen the a panel in Budapest with the id 1567917 maintained by **Smart Solar** [18]. It was registered long before 2022 and is in the same city as our institution. The consumption profile is imported from the 1000 kWh annual consumption profiles published by Hungarian distribution system operator (DSO) companies, members of MVM Group - a state-owned energy company in Hungary- computed based on statistical analytics [19]. The consumers were clustered into 4 business groups. The closest group to a school's profile was the business group #4 as its consumption was the highest during winter and the lowest during summer compared to the other groups. The price data is collected from HUPX's (Hungarian Power Exchange Company) website [20]. The chart exports contain hourly price data from the **Hungarian Power Day-ahead Market**. It means that all the pricing for each hour for the day are determined during the day before. I have chosen day-ahead market pricing as this was the closest pricing method to real-time pricing that was the only available source I could find (Intraday market prices were not considered for the present work). The first two data sources granularity is 15 minutes while the last one's is 1 hour. This difference is solved through consolidation.

The datasets have the following format listed in tables 3.2-3.4:

header	time	production_(w)
type	timestamp	float
format	"dd.MM.yyyy HH:mm"	"x.xxxx"

Table 3.2: the schema of production source file

header	Dátum	Negyedórák	Group#4
type	date	string	float
format	"dd.MM.yyyy"	"HH:mm"	"x.xxxxxx"

Table 3.3: the schema of consumption source file

header	Delivery day	Hours	Prices (EUR/MWh)
type	date	string	float
format	"dd.MM.yyyy"	"HXX HXXB"	"x.xx"

Table 3.4: the schema of price source file

In table 3.2 **time** is the timestamp of the current record with a 15 minute granularity, and **production_(w)** is the average performance of the PV system in the current time interval. In the following table 3.3 **Dátum** stands for the date while **Negyedórák** stands for the quarter hour. **Group#4** is the business group chosen from the data source, this field contains the consumption in each quarter hour in kWh. Finally in table 3.4 **Delivery day** is the date of the record, **Hours** is the hour within the date. "HXX" labels the current hour for instance 0:34 would be in "H01". HXXB refers to timeshift (CET/CEST) that occurs twice a year. **Prices (EUR/MWh)** contains the energy prices per MWh for each hour in the data set. After gathering the required csv files, the next step is to load them into Spark dataframes for transformations followed by saving the final result into Cassandra for later use. With all the transformations finished, the datasets' schemes are the following:

header	profile_id	timestamp	production_kWh
type	integer	timestamp	double
format	x	"dd.MM.yyyy HH:mm"	"x.xxxx"

Table 3.5: the schema of production dataframe

header	profile_id	timestamp	consumption_kWh
type	integer	timestamp	double
format	x	"dd.MM.yyyy HH:mm"	"x.xxxxxx"

Table 3.6: the schema of consumption dataframe

header	timestamp	price_eur_per_mwh
type	timestamp	double
format	"dd.MM.yyyy HH:mm"	"x.xx"

Table 3.7: the schema of price dataframe

The datasets are saved with write commands to Cassandra using the instance's connection. The next step is to join all these dataframes into a singular dataframe for the algorithms. Finally, the dataset is converted into a Pandas DataFrame, which supports row-wise iteration, a feature necessary for the algorithms. The schema of the final result can be seen in table 3.8:

header	timestamp	consumption_kWh	production_kWh	price_eur_per_mwh
type	timestamp	float	float	float
format	"dd.MM.yyyy HH:mm"	"x.xx"	"x.xx"	"x.xx"

Table 3.8: the schema of the final dataframe

4 IMPLEMENTATION OF THE ALGORITHMS

Alongside the three chosen algorithms I have also implemented different versions for Rule-based and LP. Additionally I have included a PV only algorithm that is used as a reference to compare the cost improvement of the algorithms that uses BESS. These algorithms are the following:

- PV only (no **BESS**)
- Greedy
- Rule based (with multiple variations)
 - using both **UT** and **LT**
 - using only **UT**
- LP solver (with multiple variations)
 - daily optimums
 - weekly optimums
 - net revenue

4.1 PV only

PV has no **BESS** and only serves to show how much savings the system can provide with the produced energy only. The strategy is simple. If the **PV** generation is greater or equal to the demand, then surplus is generated which is sold to the grid. Otherwise when the demand is greater than the production, then the difference is imported from the grid. The results can be calculated with simple Pandas Dataframe operations.

4.2 Greedy algorithm

The Greedy algorithm works similarly to the previous one, with the exception of having access to a battery system. Figure 4.1 shows the simplified activity diagram of the Greedy strategy.

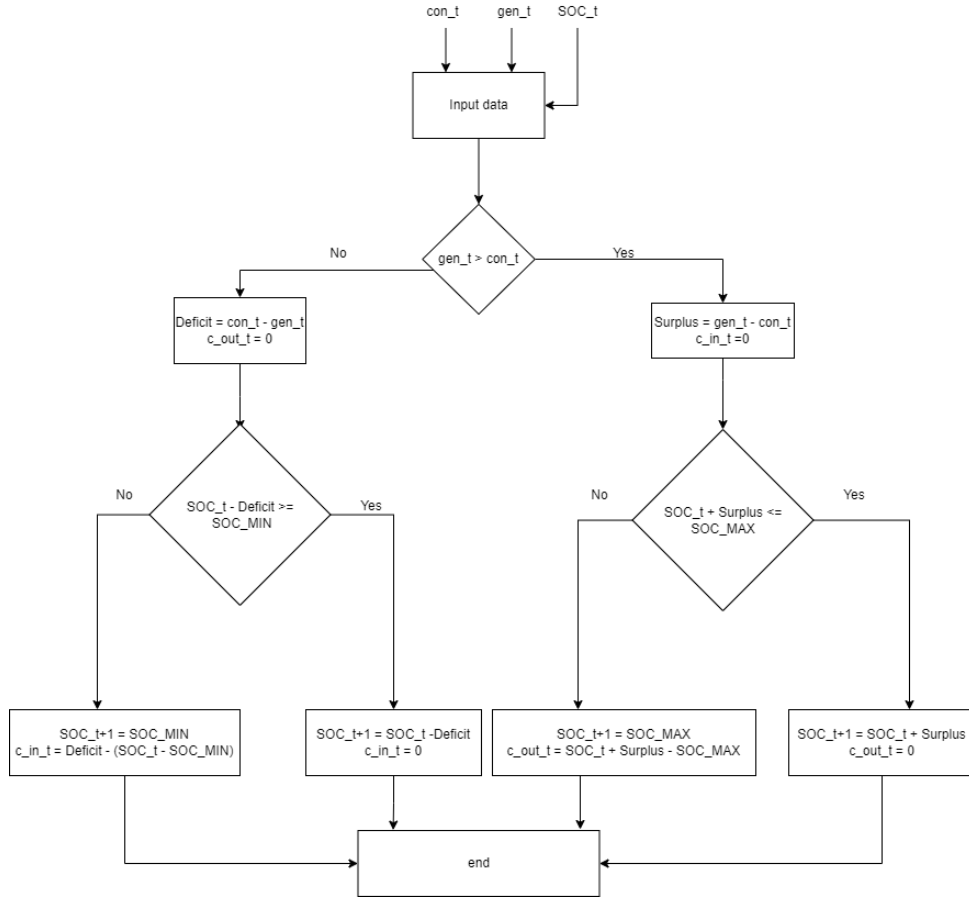


Figure 4.1: Greedy algorithm activity diagram

It is the same strategy explained in 2.4.1. **Surplus** occurs whenever $gen_t > con_t$ in the given time interval. Otherwise (when $gen_t \leq con_t$) then the algorithm branches of to the deficit side instead. Note that gen_t and con_t variables translate to **production_kwh** and **consumption_kwh** in the dataset in use respectfully.

4.3 Rule-based algorithm

The Rule-based strategy meant to be a better alternative to the Greedy strategy. It makes decisions on when to use the local energy storage system, or when to buy energy from the grid based on energy prices. It is a similar strategy found in [7]. The main difference is that in our working environment there are no different loads present, only collective demand, meaning that the demand has priority for produced energy over BESS. Only surplus energy is stored into the battery system alongside with imports from the grid. The variables and the activity diagram below that I have designed is inspired by [4]

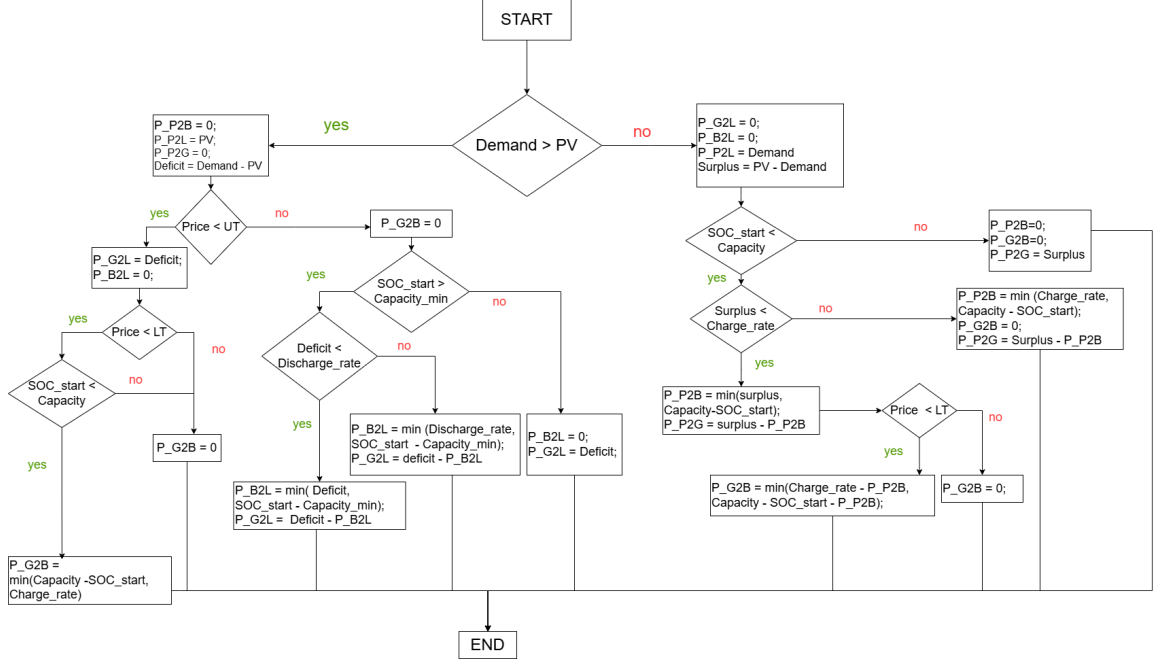


Figure 4.2: Rule-based algorithm activity diagram

4.2 Shows the logic behind the algorithm. The variables used mean the following: **P_P2L**, **P_P2B** and **P_P2L** stands for the amount of energy that is transferred from the **PV** system to the loads, battery or the grid. **P_G2L** and **P_G2B** is the energy imported from the grid directly to the loads or the **BESS**. Finally, **P_B2L** stands for the amount of energy that is used to feed the loads using the battery.

There are two main branches. If the overall demand is greater than the production, then it is the deficit side, otherwise it is the surplus side. The strategy works as explained in 2.4.2. The Upper and lower threshold are calculated as described in [7]. UT uses day-ahead market data, meaning it does not require forecast data with this setting as the hourly prices for the day are predetermined the previous day. The only variation lies in the percentiles used to calculate the different thresholds. Instead of 25th percentiles I use 10th percentiles making the algorithm more sensitive in finding the appropriate thresholds for different seasons. LT is calculated with the lower 5th percentile in all cases. The idea behind the inclusion of LT is to buy electricity when it is the cheapest for later use when overall demand might raise electricity price.

That covers the functioning of the Rule-based algorithm. Along side the basic version, that uses both thresholds, I will compare the performance of a modified Rule-based algorithm to the others. This version excludes the usage of speculative imports, meaning it would not use **P_G2B** to import cheap energy. The strategy is the same as the original algorithm, with the minor exception that it would exclude the usage of LT. That is equivalent with always branching off to the **No** side on the **LT** checks in 4.2. The goal of this comparison is to see how much it impacts the performance of this algorithm when we exclude the lower threshold for the use case where the community would rather minimise grid dependence and would avoid energy imports if possible.

4.4 Linear Programming solver algorithm

The final algorithm will utilise an open source linear programming solver, that is accessed through an interface provided by Pyomo, a modelling language. It is used to define the objective function and the constraints listed in 3.1. The parameters, consumption, production, price, starting SOC and the decision variables, ch_t , dch_t , $c_{in,t}$, $c_{out,t}$ used to define the constraint and the objective are also declared. It operates the following way: First a time period with a length of T is selected from the total dataset. Then a function takes this time period with all the production, consumption, battery starting state and price data as input parameters and produces a Pyomo model. Following that a cbc solver instance is called to solve this model. The solver finds the optimal result of the given time period and sends it back to the solver model. These results are then saved into dataframes. Each iteration uses the previous iteration's last battery state as starting point for SoC. This process repeats until all the data I provided in the test environment is processed. The whole process is illustrated in the figure 4.3 below.

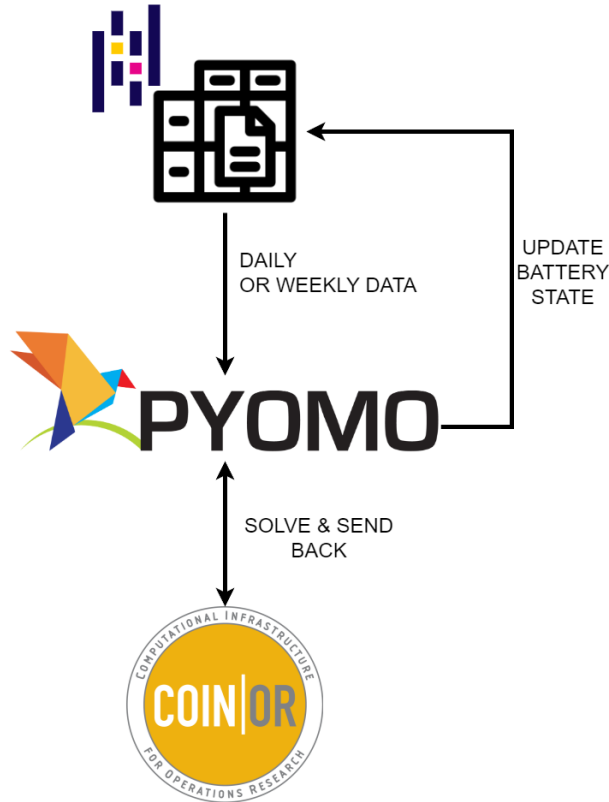


Figure 4.3: Greedy algorithm activity diagram

Note that these are only local optimums compared to the 2 year period of dataset I used in this project. Therefore I made measurements using different time intervals: a daily and a weekly version. Additionally I have tested this algorithm's capabilities with an alternative objective function:

$$\text{Minimise } \sum_t^T c_{in,t} * p_t - \sum_t^T c_{out,t} * p_t \quad (4.1)$$

This objective takes the difference of the purchased and sold energy, meaning the new objective is optimising the net revenue. This version encourages the solver to store energy not only for saving but for commercial purposes (daily arbitrage) as well. As it would purchase electricity in bulk to sell it back to the grid for profit.

4.5 Operation

To summarize the operation of the algorithms, the entire time period is divided into 15-minute intervals. The algorithms make decisions at each interval, represented by decision variables outlined in Section 3. These decisions must adhere to the constraints detailed in Section 3.1. The example below shows a subset of the decisions made by the LP algorithm, illustrating its operation.

ROW	gen_t	con_t	$price_t$	soc_t	ch_t	dch_t	$c_{in,t}$	$c_{out,t}$	COST	REV
0	0.00	0.18	113.19	0.00	0.00	0.00	0.18	0.00	0.02	0.00
1	0.00	0.18	113.19	0.00	0.00	0.00	0.18	0.00	0.02	0.00
2	0.00	0.18	113.19	0.00	0.00	0.00	0.18	0.00	0.02	0.00
3	0.00	0.18	113.19	0.00	0.00	0.00	0.18	0.00	0.02	0.00
4	0.00	0.18	105.06	0.00	0.00	0.00	0.18	0.00	0.02	0.00
.....										
15	0.00	0.18	94.05	4.48	4.48	0.00	4.65	0.00	0.44	0.00
16	0.00	0.18	100.03	4.30	0.00	0.18	0.00	0.00	0.00	0.00
17	0.00	0.18	100.03	4.11	0.00	0.18	0.00	0.00	0.00	0.00
18	0.00	0.18	100.03	3.93	0.00	0.18	0.00	0.00	0.00	0.00
19	0.00	0.19	100.03	3.75	0.00	0.19	0.00	0.00	0.00	0.00

Table 4.9: LP operation

In each row, con_t has to be satisfied using local or external sources. In the first few rows, there was no option other than importing all the required energy using $c_{in,t}$. The **COST** was calculated as $price_t \times 0.001 \times cin_t$, since the unit price is measured in euros per MWh. By the 15th row, the algorithm decided to import more energy than required and stored the surplus in the battery energy storage system (BESS). The soc_t variable is equal to the previous state, plus the charged energy, minus the discharged amount in the current iteration ($soc_t = soc_{t-1} + ch_t - dch_t$). Subsequently, the battery was used in the following hour as $price_t$ increased, utilizing the stored energy for cost optimization.

4.6 Testing

To validate my implementations of the three algorithms: Greedy, Rule-based and LP in the working environment, I have created a test dataset with 10 records of production, consumption and prices. This can be seen in table 4.10:

Consumption (kWh)	Production (kWh)	Price (€/MWh)
15	8	-3.0
10	6	1.0
8	10	-2.0
8	10	-1.0
20	10	-0.5
30	7	5.0
14	9	7.5
10	8	15.0
25	11	9.0
13	10	8.5

Table 4.10: Test Data for validation

I have included negative price values as they can occur in real-world scenarios. These values play a critical role in decision-making and serve as a valuable test for evaluating the proper functionality of the Linear Programming (LP) algorithm.

Testing the Greedy Algorithm: The Greedy algorithm was evaluated to ensure it maintains consistency in the battery's state of charge based on its decision variables, and to verify that its logic operates as intended. To facilitate this testing, the dataset was extended with manually calculated decision variables (c_{in} , c_{out} , ch , and dch) and state-of-charge values that represent the expected outputs if the algorithm functions correctly. The dataset was then processed using the Greedy algorithm, and the results were compared against the expected values for each row. The outcome of this comparison is presented in the accompanying tables 4.11-4.13:

test_SOC_start	SOC_start	test_SOC_end	SOC_end
0	0.0	0	0.0
0	0.0	0	0.0
0	0.0	2	2.0
2	2.0	4	4.0
4	4.0	0	0.0
0	0.0	0	0.0
0	0.0	0	0.0
0	0.0	0	0.0
0	0.0	0	0.0
0	0.0	0	0.0

Table 4.11: SOC_related columns for Greedy

test_ch	ch	test_dch	dch	test_cin	c_in	test_cout	c_out
0	0.0	0	0.0	7	7.0	0	0.0
0	0.0	0	0.0	4	4.0	0	0.0
2	2.0	0	0.0	0	0.0	0	0.0
2	2.0	0	0.0	0	0.0	0	0.0
0	0.0	4	4.0	6	6.0	0	0.0
0	0.0	0	0.0	23	23.0	0	0.0
0	0.0	0	0.0	5	5.0	0	0.0
0	0.0	0	0.0	2	2.0	0	0.0
0	0.0	0	0.0	14	14.0	0	0.0
0	0.0	0	0.0	3	3.0	0	0.0

Table 4.12: Columns related to charging, discharging, and energy flows for greedy

test_selling_price	sell_price	test_buy_price	buy_price
-0.0210	-0.0210	-0.0210	-0.0210
0.0040	0.0040	0.0040	0.0040
0.0000	0.0000	-0.0000	-0.0000
0.0000	0.0000	-0.0000	-0.0000
-0.0030	-0.0030	-0.0030	-0.0030
0.1150	0.1150	0.1150	0.1150
0.0375	0.0375	0.0375	0.0375
0.0300	0.0300	0.0300	0.0300
0.1260	0.1260	0.1260	0.1260
0.0255	0.0255	0.0255	0.0255

Table 4.13: Columns related to energy pricing for greedy

The actual values match the test equivalents, meaning that Greedy works accordingly.

Testing Rule-based: I have evaluated the Rule-based algorithm similarly to Greedy with the same method. The results can be seen in tables 4.14 and 4.15. We can see that just as with the case for Greedy, Rule-based works as intended with no errors.

test_SOC_start	SOC_start	test_SOC_end	SOC_end
0.0	0.0	12.5	12.5
12.5	12.5	25.0	25.0
25.0	25.0	37.5	37.5
37.5	37.5	50.0	50.0
50.0	50.0	50.0	50.0
50.0	50.0	37.5	37.5
37.5	37.5	32.5	32.5
32.5	32.5	30.5	30.5
30.5	30.5	18.0	18.0
18.0	18.0	15.0	15.0

Table 4.14: SOC data of Rule based

test_buy_price	buy_price	test_sell_price	sell_price
-0.0585	-0.0585	0	-0.0
0.0165	0.0165	0	0.0
-0.0210	-0.0210	0	-0.0
-0.0105	-0.0105	0	-0.0
-0.0050	-0.0050	0	-0.0
0.0525	0.0525	0	0.0
0.0000	0.0000	0	0.0
0.0000	0.0000	0	0.0
0.0135	0.0135	0	0.0
0.0000	0.0000	0	0.0

Table 4.15: Columns related to energy pricing for Rule-based

Testing LP: Finally I have evaluated LP, by checking if the output variables it created abides to the rules defined using pyomo.

con	prod	price	soc	ch	dch	c_in	c_out	y_ch	y_dch	y_cin	y_cout
15	8	-3.0	12.5	12.5	0.0	19.5	0.0	1.0	0.0	1.0	0.0
10	6	1.0	8.5	0.0	4.0	0.0	0.0	0.0	0.0	0.0	0.0
8	10	-2.0	21.0	12.5	0.0	10.5	0.0	1.0	0.0	1.0	0.0
8	10	-1.0	33.5	12.5	0.0	10.5	0.0	1.0	0.0	1.0	0.0
20	10	-0.5	46.0	12.5	0.0	22.5	0.0	1.0	0.0	1.0	0.0
30	7	5.0	33.5	0.0	12.5	10.5	0.0	0.0	1.0	1.0	0.0
14	9	7.5	28.5	0.0	5.0	0.0	0.0	0.0	1.0	0.0	0.0
10	8	15.0	26.5	0.0	2.0	0.0	0.0	0.0	1.0	0.0	0.0
25	11	9.0	14.0	0.0	12.5	1.5	0.0	0.0	1.0	1.0	0.0
13	10	8.5	11.0	0.0	3.0	0.0	0.0	0.0	1.0	0.0	0.0

Table 4.16: LP test data

According to Table 4.16, all constraints, including those related to charge, discharge, state of charge, and the balance rule, have been satisfied. However, additional variables, namely: y_{ch} , y_{dch} , y_{cin} , and y_{cout} , are also present. These are binary support variables used in constraints 3.4 and 3.9, respectively. Since linear programming cannot handle the product of two variables, binary variables were introduced instead. The condition defined was that the sum of y_{ch} and y_{dch} , as well as the sum of y_{cin} and y_{cout} , must not exceed 1. This ensures that only one of the variables can be 1 at a time. The test results confirm that this condition holds true across all records.

5 RESULTS

The algorithms have been tested in 5 different cases. In each case I have re-scaled the production data by changing the PV systems' installed capacity (peak power, theoretical peak output). The original peak was **10.8 kWp**. Each row has been multiplied by the ratio of the new peak performance to the original one. While the consumption data has been scaled up to 10000 kWh annual consumption in all cases. That means a total of 20000 kWh consumption for the whole 2 year time interval. The table below summarises all the test cases:

Cases (PV capacity kWp)	20	8	6	4	2
PV production (kWh)	52612.7	21045.08	15783.81	10522.54	5261.27
gen/con ratio	2.63	1.05	0.79	0.52	0.26

Table 5.17: The different test cases

Gen/con ratio stands for the ratio of the total amount of production to the total amount of consumption. With this I have created scenarios that ranges from overproduction that is twice as large as the consumption to severe underproduction where the demand is nearly four times larger than the production. My goal is to see what differences does it make in the performance of the algorithms.

5.1 Rule based hyper parameter tuning

Going back to the Rule-based algorithm, where I have stated that different percentiles can be used for different seasons. Therefore there are percentile variations that work better than others with the dataset. I used a heuristic approach to find the best variations for each scenarios at least based on the tests. I took the average production of each season of the dataset, then chose a week within each season that has the closest average production to the seasonal average.

With this I have created a four week long test dataset that includes weeks from all of the four seasons. With that dataset I tested all possible variations of the percentiles between 0.1 and 0.8. I left out 0.9 because of hardware limitations, this way there are less variations that has to be processed. I ordered the results based on the energy cost, then I selected variations from the top 20 results that I have tested on the original data.

Finally I have chosen the best performing variations based on the top results of the four-week-tests for each scenario. Both versions of the Rule-based strategy have been tested. The first version with both UT and LT tend to perform the best with the minimum allowed percentiles which is 0.1 for all seasons. The reason behind it might be the fact, that it is more profitable if the system uses up the energy therefore freeing up the capacity for the energy import with cheap grid prices. The second version however does not use low energy price purchases for optimisation purposes, therefore it has to spare the available energy to be more cost efficient. This can be seen in table 5.18.

scale factor	Winter	Spring	Summer	Autumn
20	0.1	0.1	0.1	0.1
8	0.4	0.1	0.1	0.3
6	0.7	0.1	0.1	0.6
4	0.8	0.1	0.1	0.8
2	0.9	0.8	0.6	0.9

Table 5.18: Percentiles for Rule-based strategy with no lower threshold for each scenario

Notice that the less the energy production, the higher percentiles are used by the second version. I will use these percentiles for comparisons.

5.2 Comparison of all strategies

I will compare all the algorithms using the aforementioned indicators at 3.1. The comparison will include two versions of the Linear Programming, the two versions of the Rule-based algorithms, the Greedy strategy as well as the PV only case for reference. I will limit the total comparison of these algorithms to two scenarios: the 8 kWp and the 6 kWp case.

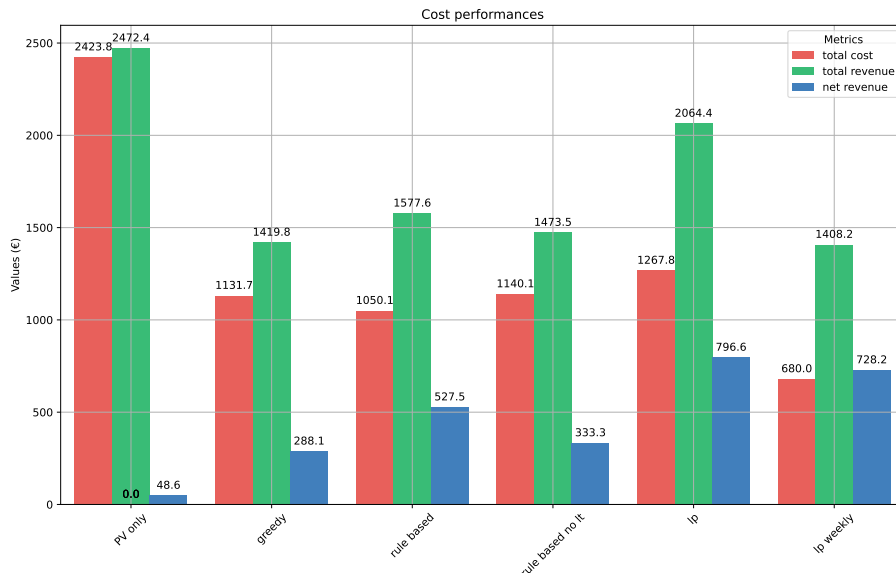


Figure 5.1: Cost, revenue and net revenue of all the strategies with 8 kWp

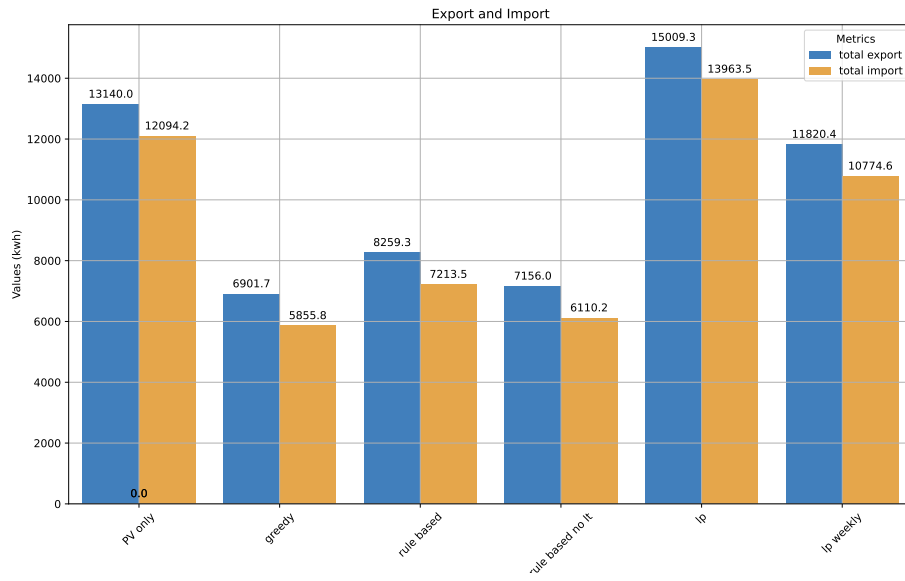


Figure 5.2: Import and export of all the strategies with 8 kWp

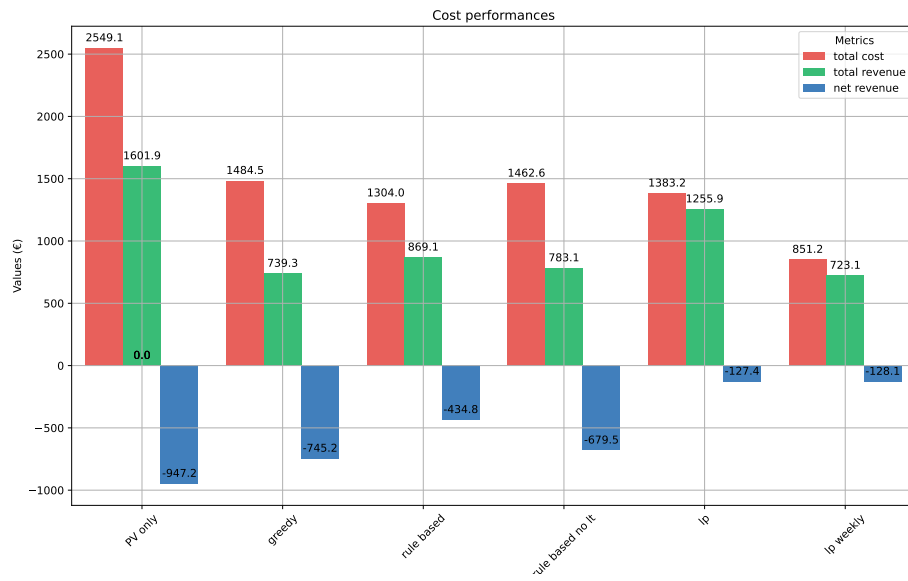


Figure 5.3: Cost, revenue and net revenue of all the strategies with 6 kWp

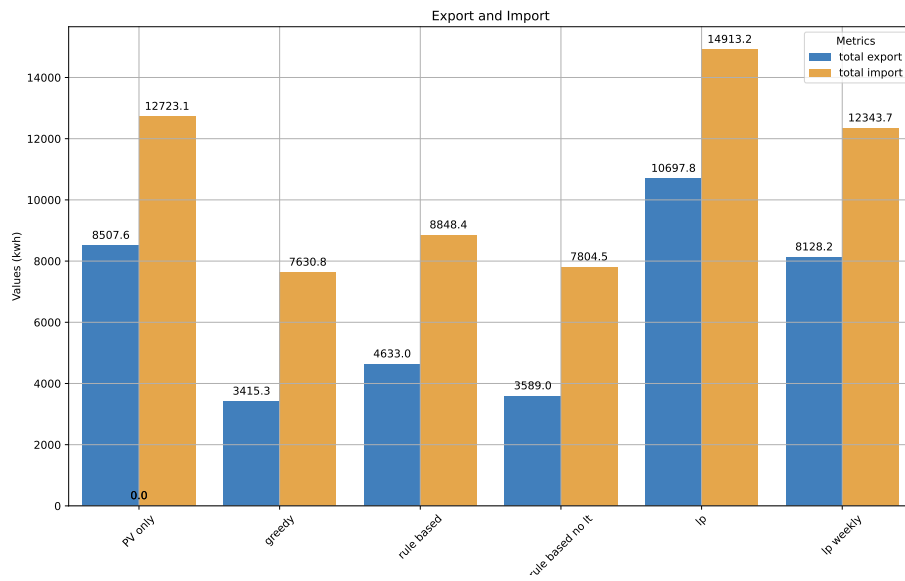


Figure 5.4: Import and export of all the strategies with 6 kWp

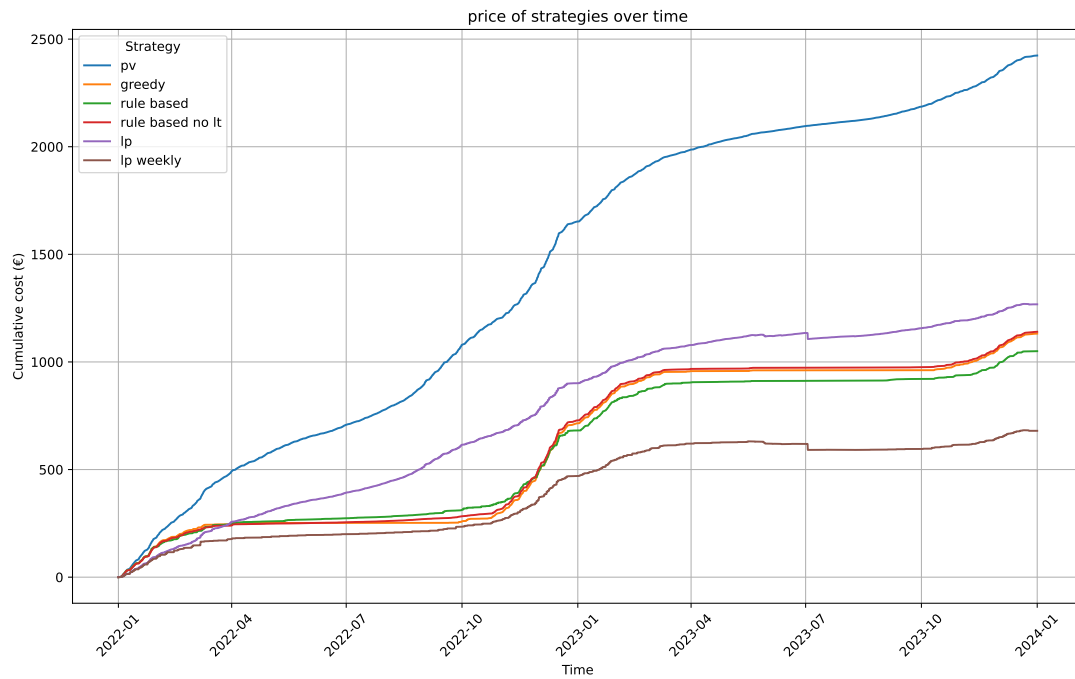


Figure 5.5: The cumulative energy cost of the strategies over time with 8 kWp

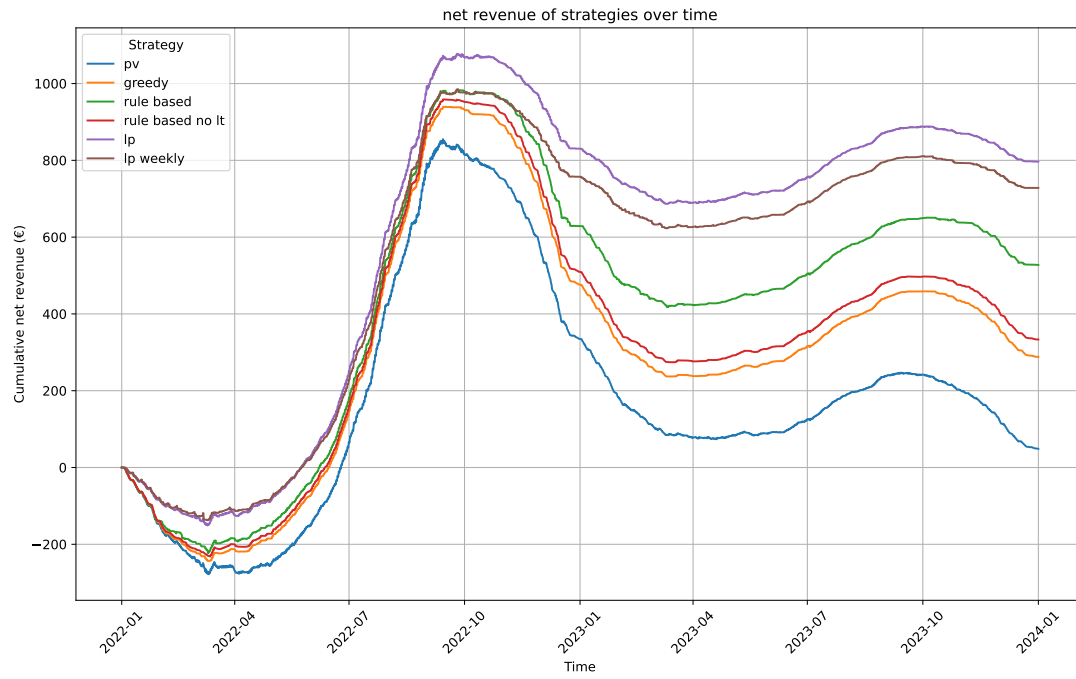


Figure 5.6: The cumulative net revenue produced by the strategies over time with 8 kWp

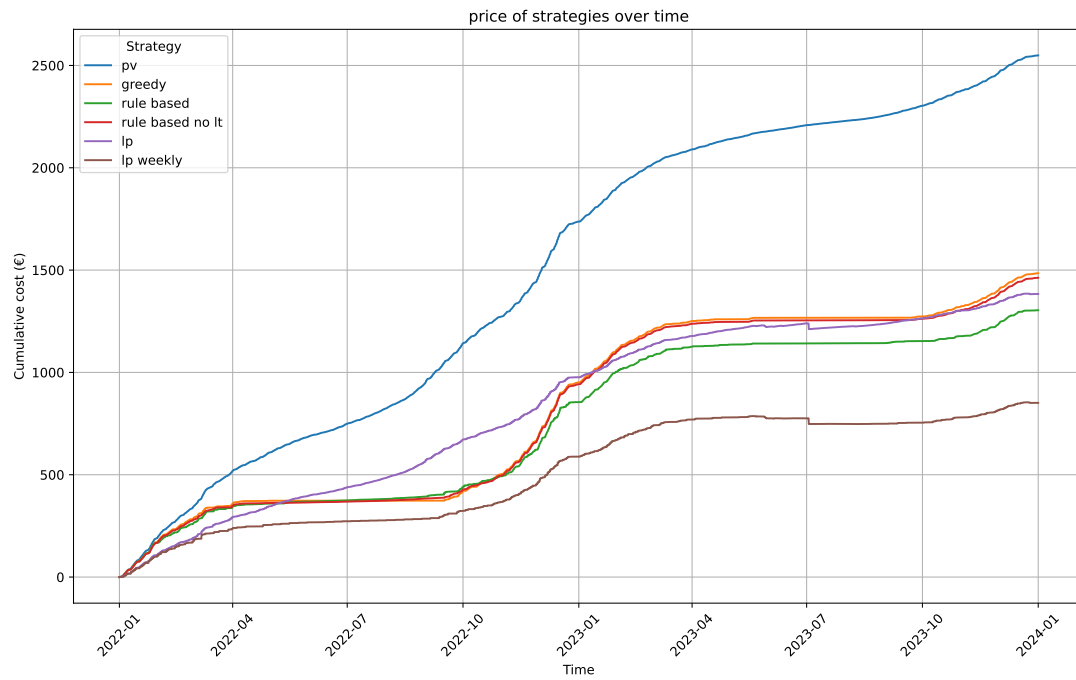


Figure 5.7: The cumulative energy cost of the strategies over time with 6 kWp

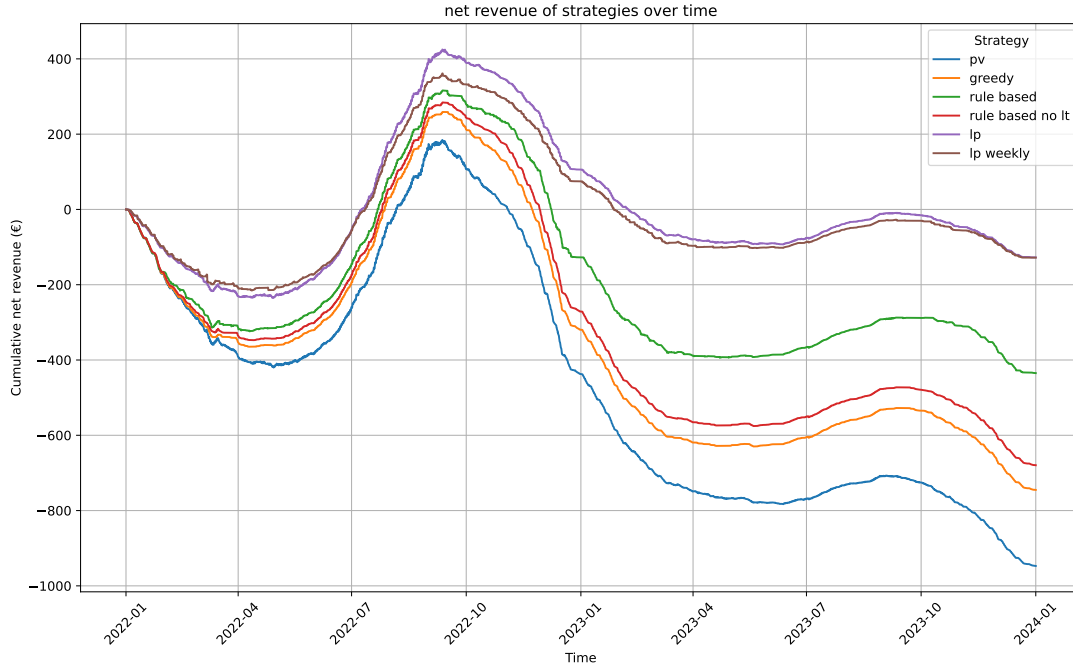


Figure 5.8: The cumulative net revenue produced by the strategies over time with 6 kWp

These figures show an important fact regarding the efficiency of the LP strategy. Namely on figures 5.1 and 5.3, if we omit the weekly version, then the best performing strategy by far is the Rule-based algorithm in both cases, outperforming the Linear Programming algorithm. My explanation for this is that the first version of LP determines daily forecast. This results in daily local optimums that divides the whole two year period into more then seven hundred separate local optimisation problems. Even if we found an optimal solution for today, the best solution might change if we would have access to tomorrow's data. Consider figures 5.5 and 5.7. During the first few months in both cases the daily version of Linear Programming outperforms the rest of the algorithms. The change in performance happened after April or May depending on the scenario. That means that even if the strategy performs better in short term, daily optimums are not sufficient for longer terms such as yearly periods.

By increasing the interval from daily to weekly optimums, the algorithm can make a larger impact on performance. The previously mentioned figures show that the weekly version of LP has significantly better cost reductions compared to all of the other algorithms for the entire two year period. However the increase of the interval of the search space demands better performing forecast algorithms. Additionally I have included figure 5.9 to see how much improvement can the Linear Programming make with the best energy production scenario. As we can see the daily version underperforms with 1000 € cost compared to the Greedy strategy's performance with only 418 €. The weekly version however still could improve the energy cost with around 100 € difference, however this gap is considerably smaller than in the rest of the scenarios.

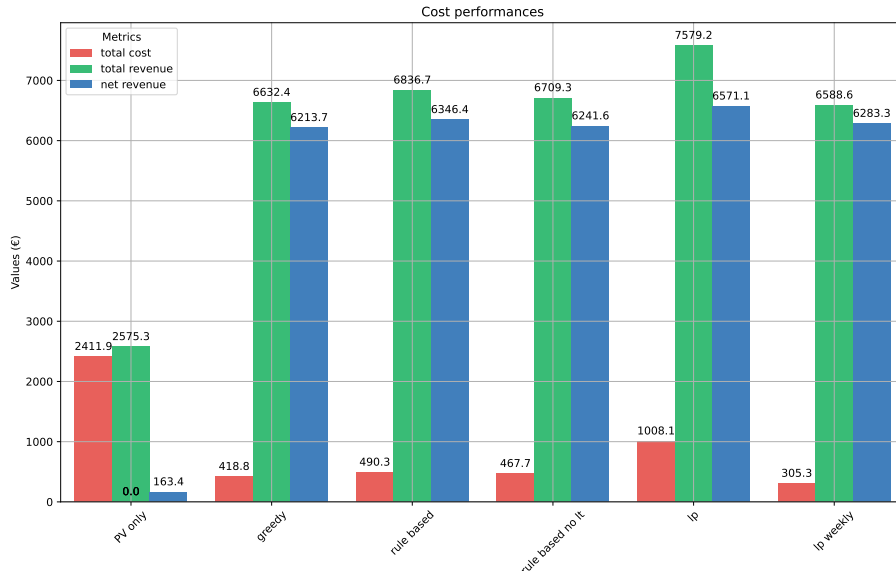


Figure 5.9: Cost, revenue and net revenue of all the strategies with 20 kWp

Considering the import and export amounts of the algorithms, it is clear that LP depends the most on outside sources. The daily version even has higher import/export traffic than the PV only solution. As a result of greater export output and the cost optimisation of both LP algorithms they achieved significantly better net revenue compared to the rest of the strategies, even though they only optimise on energy cost.

I have also included figures that show the change of the state of charge of the battery showing how the different algorithms utilise the available capacities over time. 5.10 and 5.11 displays the SOC of BESS in winter and summer time using data from 2022 January and July. Additionally figures 5.12, 5.13 and 5.14 show the strategies imports and exports through January and July. Imports are represented with solid while exports represented with dashed lines.

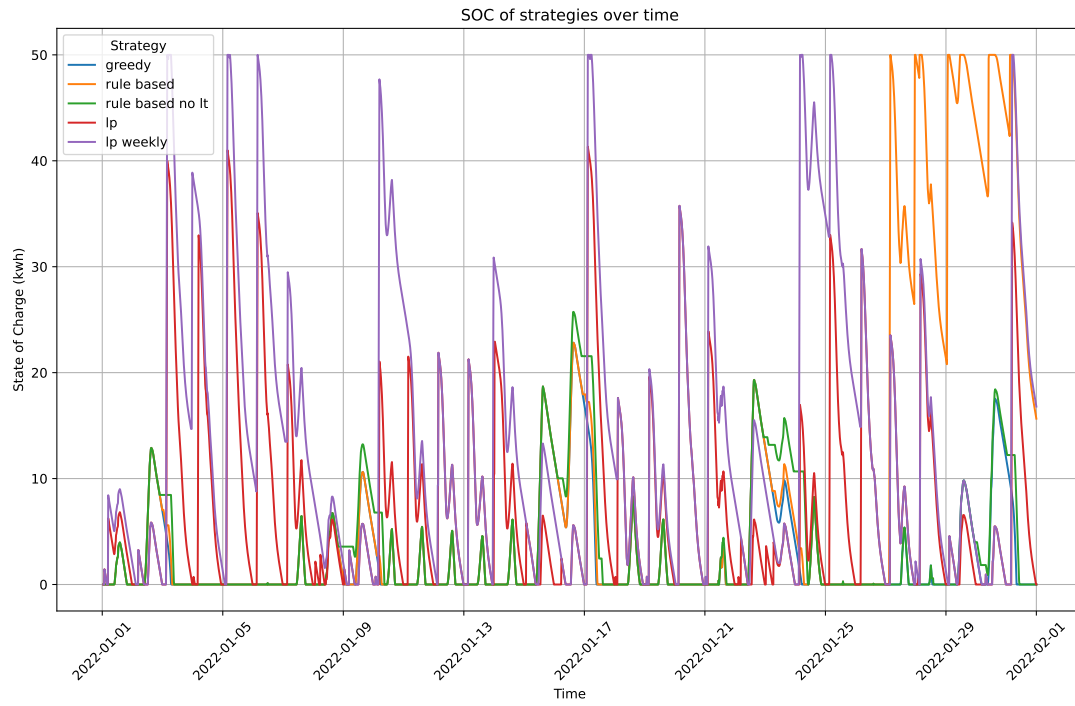


Figure 5.10: SOC in January 2022 with 8 kWp

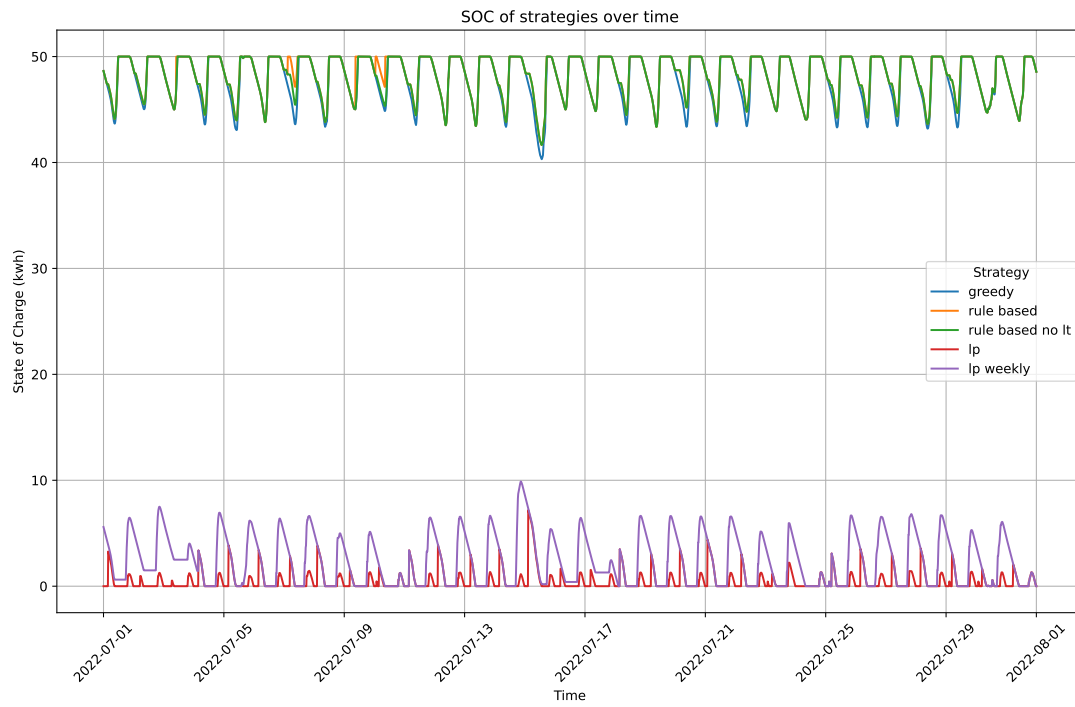


Figure 5.11: SOC in July 2022 with 8 kWp

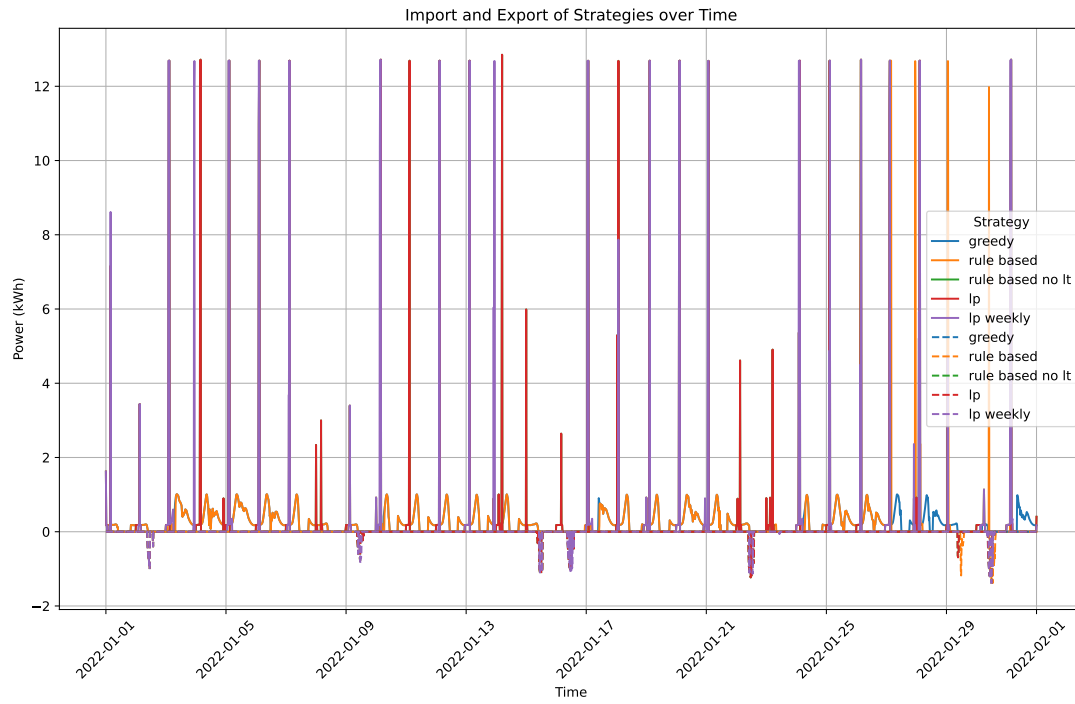


Figure 5.12: import and export of strategies January 2022



Figure 5.13: import and export of strategies July 2022

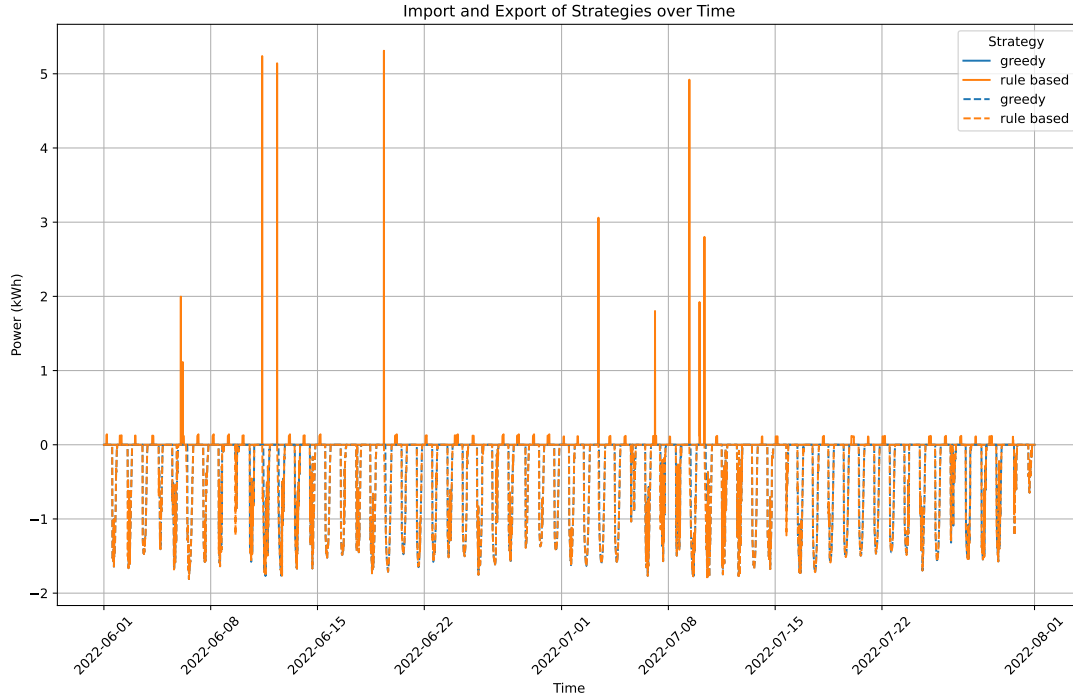


Figure 5.14: import and export of Rule-based and Greedy strategies

In figures 5.12, 5.13 and 5.14 the imported amounts are represented by continuous lines, while the exported amounts are illustrated using dashed lines. As we can see, depending on the season the algorithms work differently. First during winter all of the algorithms deplete the battery periodically as there is energy shortage. The Rule-based and the LP algorithms have significantly higher charge spikes due to the additional imports they made. On the other hand the no LT version and the Greedy algorithm have lower state of charge over the course of the month. During summer another interesting difference can be observed between LP and the other algorithms' behaviours. While the basic strategies do not consider selling the excess energy which makes the battery's almost always fully charge, the Linear Programming algorithms has their batteries depleted leaving only as much energy in the system as needed.

5.3 Performance comparison using varying time intervals

After the previous observations regarding the performance difference between the two versions of LP the question arises that how much cost improvement can be achieved by increasing the interval of the local optimums. In a separate experiment I have compared LP results using 1, 2, 5, 7, 10, 14 and 28 day intervals:

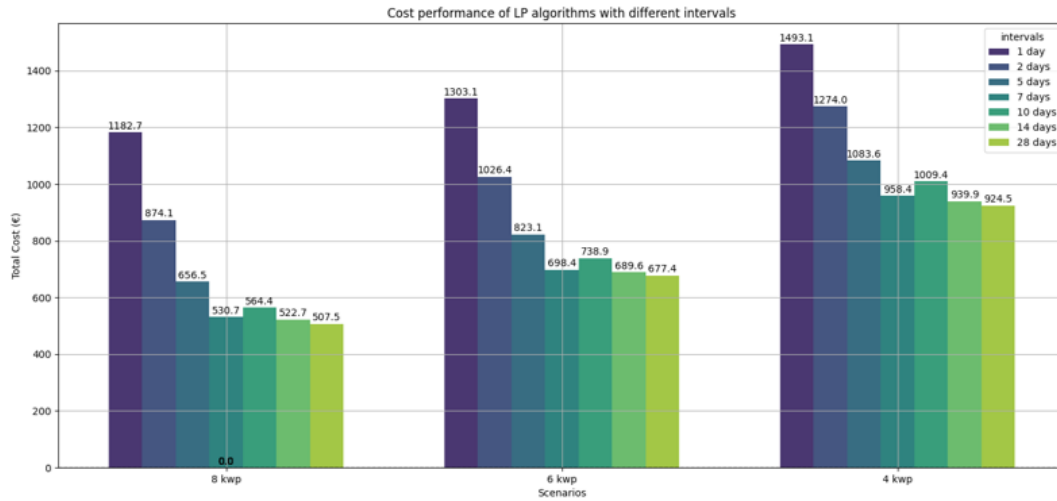


Figure 5.15: LP Cost Performance Improvement with Increasing Intervals

The experiment was repeated in three different scenarios: 8, 6, and 4 kWp setups reused from previous tests. It is clear that as the interval increases, cost performance improves (with the exception of the 10-day interval). In the first three versions—1, 2, and 5-day intervals—a significant cost reduction is observed at each step. However, after the 1-week interval, the reduction nearly stagnates. It appears that around the 1-week interval, based on the two years of data analyzed, further cost reductions cannot be achieved. Additionally, as PV production decreases, the difference in cost between the best and worst-performing versions of the Linear Programming (LP) models also diminishes. At 8 kWp, this difference is **675.2 kWh**, while at 6 and 4 kWp it is **625.7** and **568.6 kWh**, respectively. Based on this experiment, it is safe to conclude that while daily forecasting is inefficient, the weekly approach explored in this study proves to be an optimal choice, effectively maximising the performance of LP.

5.4 Using Linear Programming for Net Revenue Optimisation

Last but not least I have tried out the LP with a different objective function that includes the total revenue defined in 4.1 I have compared the net revenue, the total export and import made by this version of the daily LP algorithm. Figures 5.16 and 5.17 show that by far LP has the largest amount of net revenue as well as imports and exports. It has an overwhelming advantage over the rest of the strategies even with daily forecasting. This however comes with a cost. Figures 5.18 and 5.19 show that this strategy uses the battery more often having charge and discharge spikes very frequently. On paper it is the best option to optimise net revenue along side the rest of the strategies, however in practise this type of usage can shorten battery life [16], therefore increasing the overall cost of the system as the community would have to replace the battery unit more often.

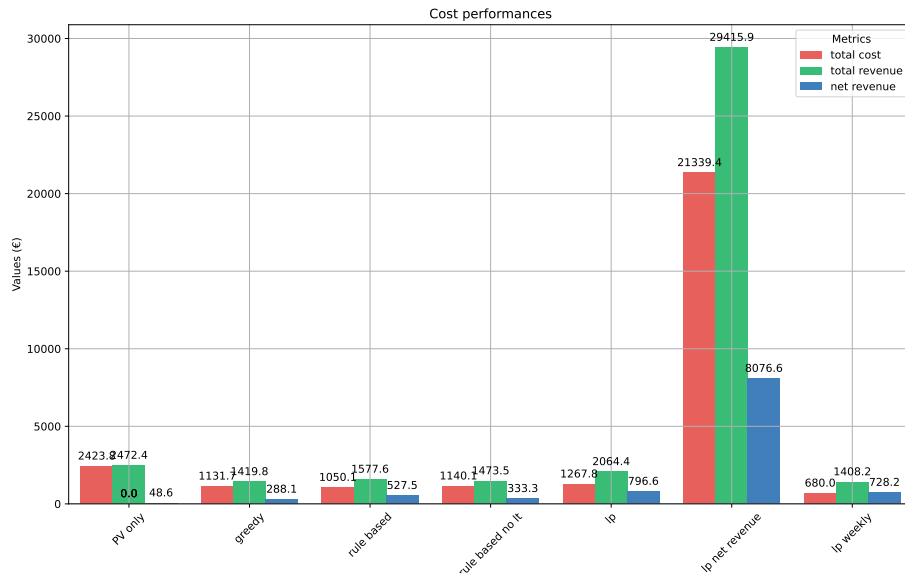


Figure 5.16: Total cost of strategies including net revenue LP with 8 kWp

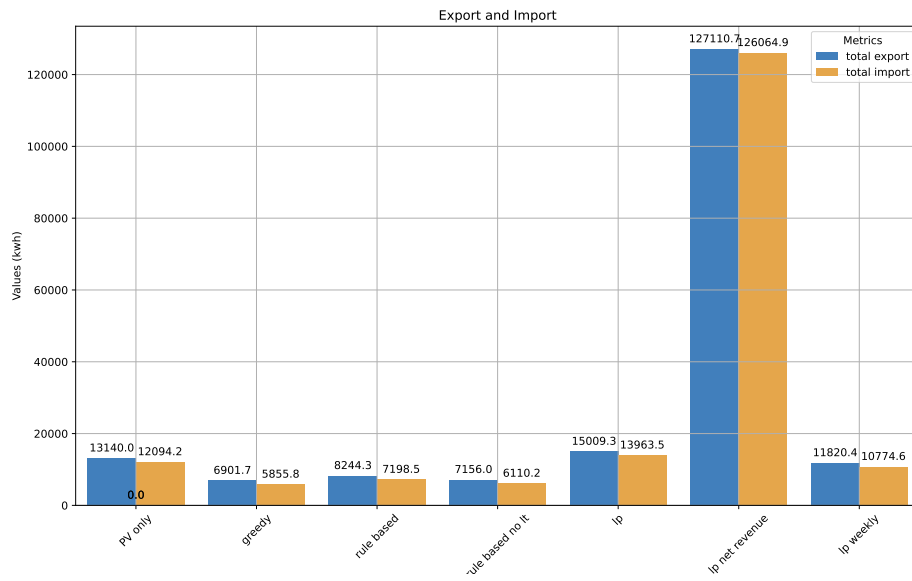


Figure 5.17: Total export and import of strategies including net revenue LP with 8 kWp

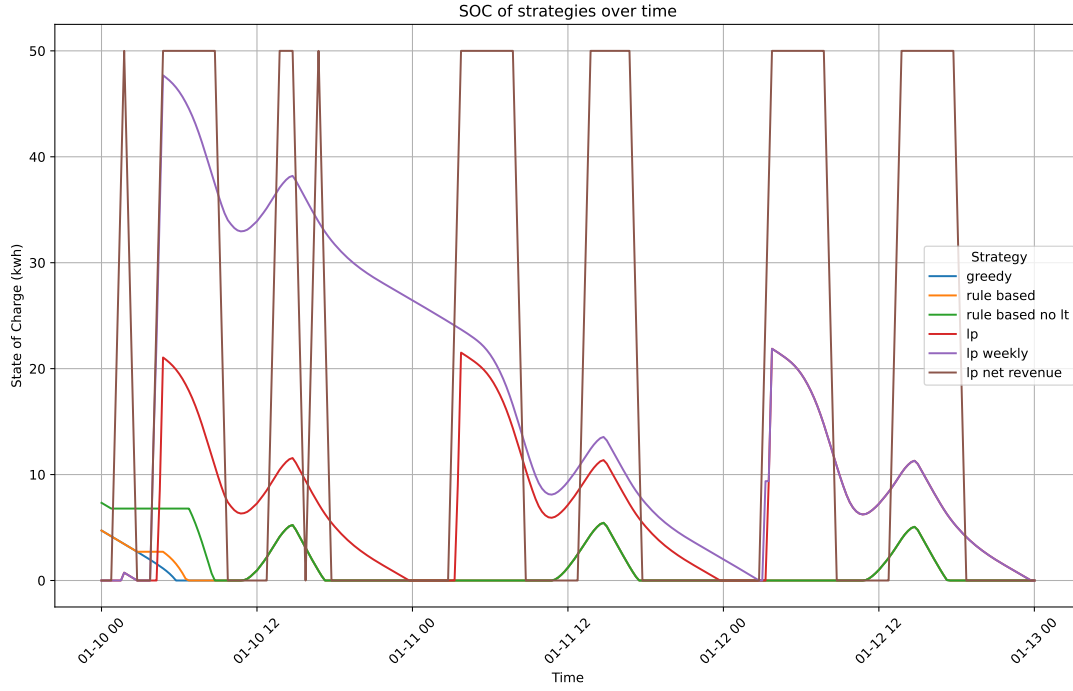


Figure 5.18: Comparing SOC of net revenue LP to the other strategies January 2022

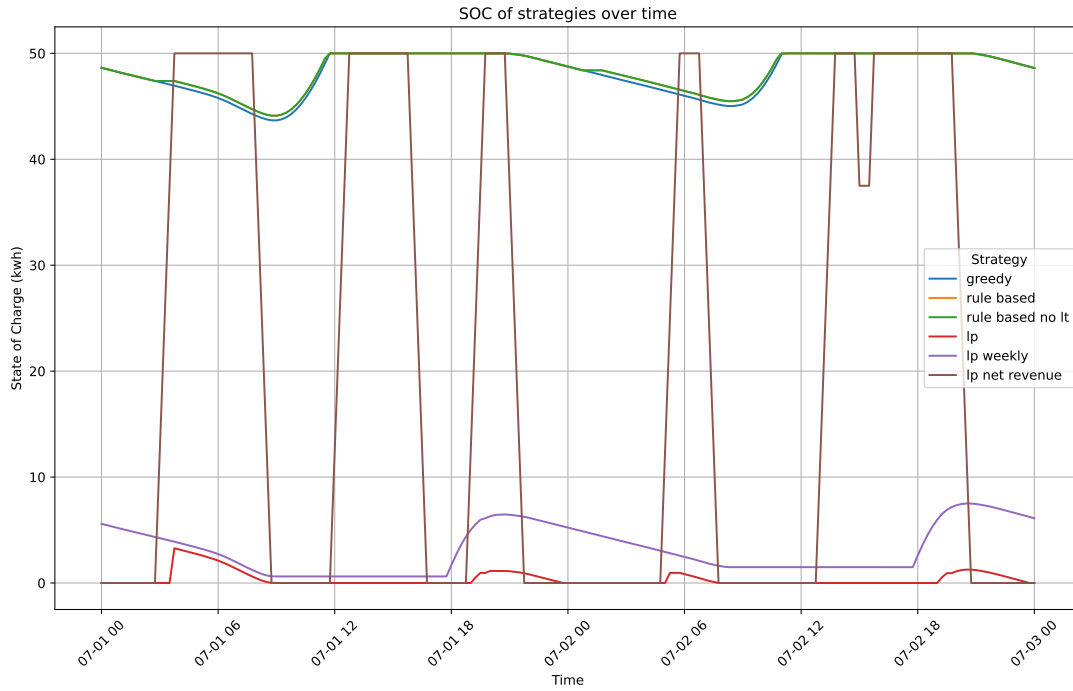


Figure 5.19: Comparing SOC of net revenue LP to the other strategies July 2022

5.5 Comparison of Greedy and Rule-based algorithm

Finally I would revisit the Greedy and both versions of Rule-Based algorithms to compare their output in the different test cases. Consider figure 5.20. It shows the total cost of the Greedy

and the two versions of the Rule-based algorithm for each scenario. The scenarios are labeled with the **scale_factor** in use, which is the peak kW performance of the system. Two main considerations can be made. As the peak performance decreases, the greater the difference becomes in costs between the original Rule-based and the Greedy strategies. An additional figure 5.23 shows the cost improvement in percent compared to the Greedy strategy for both algorithms. While the first version of Rule-based strategy performs significantly better in most cases, the version without LT still has better results as well, however the difference it makes compared to Greedy is negligible. This version has closely the same characteristics as the Greedy strategy. The usage of cheap energy purchase clearly has an advantage for the Rule-based strategy.

The second consideration is that depending on the scenario, there are cases where Rule-based cannot outperform Greedy or even has worse results. The first case with 20 kWp shows that clearly. The reason behind it is that if a community has a system that can produce multiple the amount of energy needed annually then saving it becomes less necessary. It is more efficient if they use as much energy as they can instead. These results coincide with an observation in Table 5.18. The best percentile variations for higher production are smaller values, making the algorithm save up less energy. With the increase of production, the percentiles converge to zero, or in other word, the Greedy strategy. Even if the UT percentiles for the first version of the Rule-based strategy are disabled, the cost would still be around **443 €** as it uses cheap energy purchases. However as the system produces more than enough energy these purchases are unnecessary and just create additional costs.

In case the community could sell its excess energy for the actual real-time price, then theoretically the algorithms would produce revenues shown in 5.21. We can observe that even if the difference in income is small, the Rule-based strategy is consistently the best compared to the other two. Overall, the reduction in production is accompanied by a decrease in revenue, as expected. On figure 5.22 the net revenue is shown. Even though there was a case in which the Rule-based strategy fell behind the Greedy strategy considering energy costs, in net revenue it outperformed all the other strategies. The differences however are getting smaller with the increase of production.

These results are based on the fact that the community is allowed to sell their energy to the grid for the actual price. That might not be the case for all communities. Therefore the primary focus of this experiment is the reduction of energy cost defined as the original objective. Based on that the conclusion is that for overproduction a simple Greedy strategy is sufficient and might be the best option between the two. On the other hand, if production is scarce compared to consumption, then the Rule-based algorithm should be used. Additionally, the performance improvement of the no-LT version of the Rule-based strategy has no impact compared to the Greedy approach, making its use redundant.

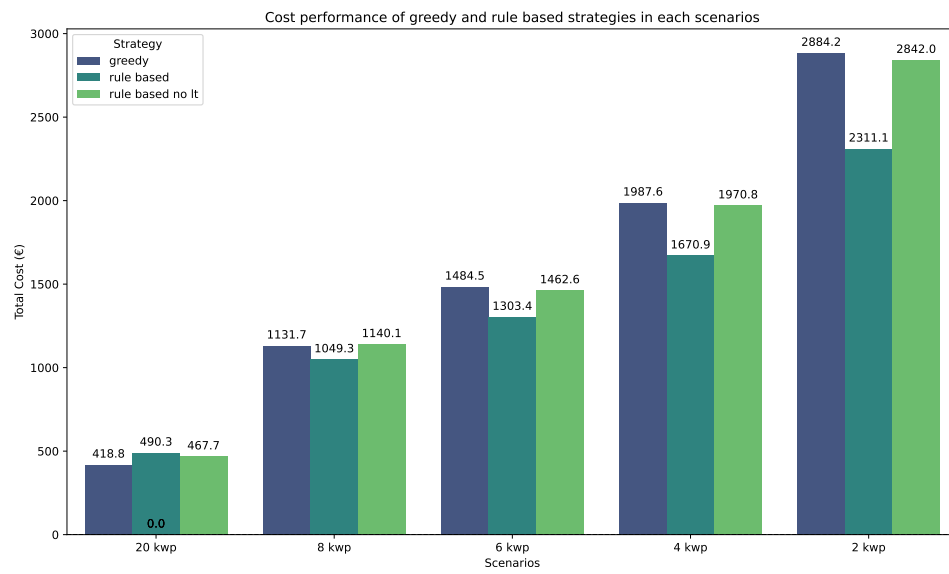


Figure 5.20: The cost performance of Greedy and Rule based algorithms

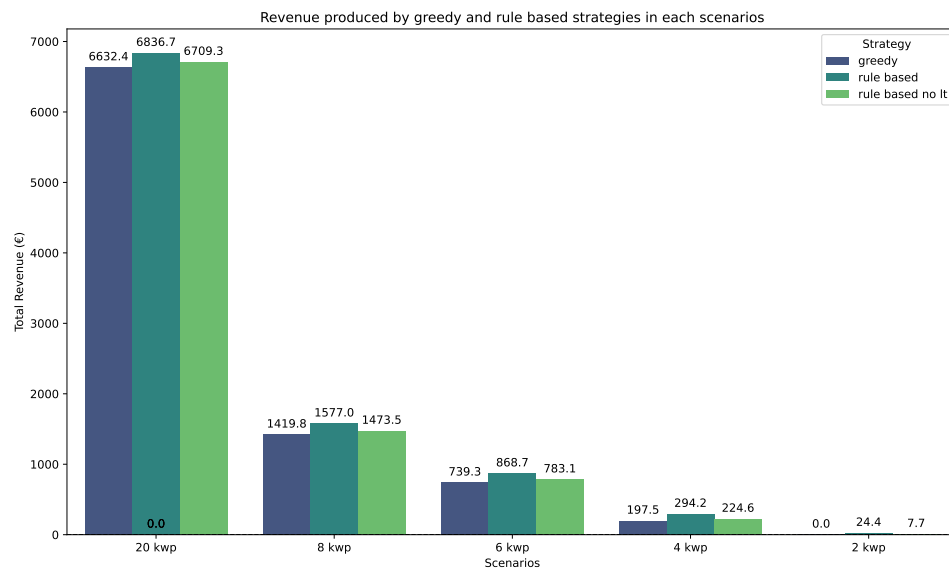


Figure 5.21: The total revenue produced by Greedy and Rule based algorithms

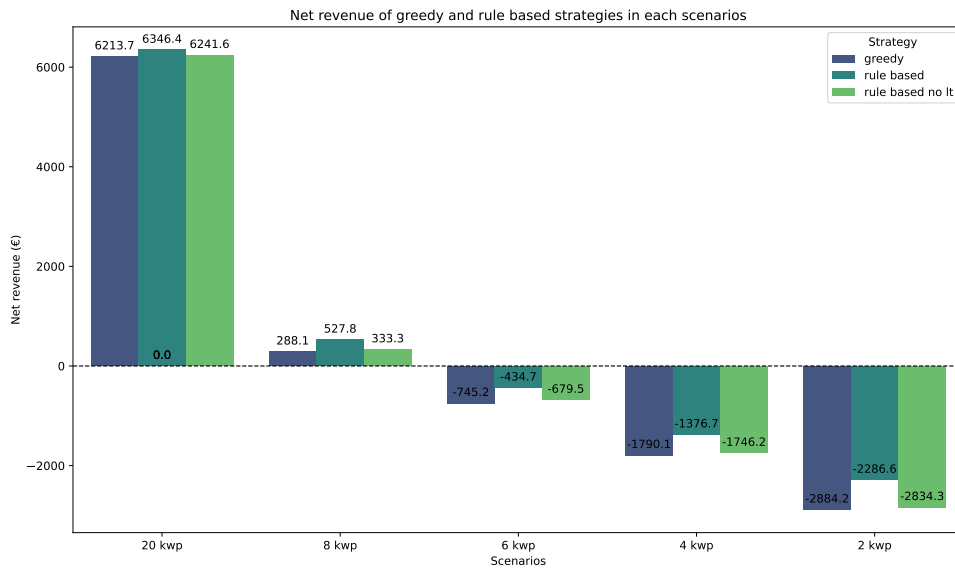


Figure 5.22: Net revenue of Greedy and Rule based algorithms

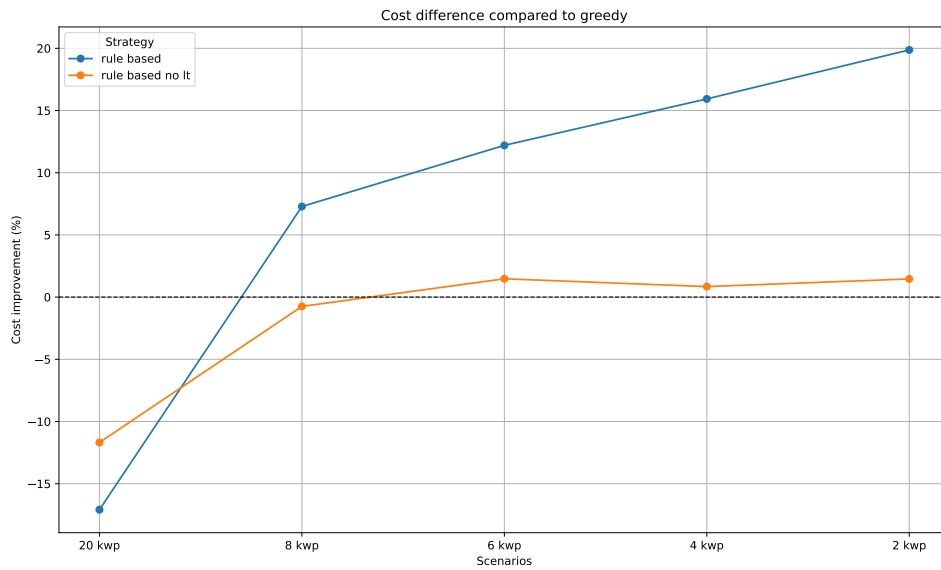


Figure 5.23: Performance difference compared to the Greedy algorithm in percent

5.6 Overall comparison

To evaluate the results, I have included additional figures that highlight the cost performance and the export-import amounts in both the best and worst test cases. The second part of Figure 5.24 demonstrates that, in the worst test case, the daily version of LP outperforms the non-LP algorithms in terms of cost, unlike in the best case with 20 kWp. However, based on previous

observations, the daily version of LP generally proves inefficient for cost optimisation compared to its counterparts. The same holds true for the no LT version of the Rule-based algorithm, which yields similar results to the Greedy version across all test cases, rendering it redundant when compared to the original Rule-based approach. As a result, both the daily LP and the no LT Rule-based algorithm have been excluded from further consideration. Additionally, while the LP algorithm optimising for net revenue shows potential, it is not currently viable for real-world implementation, as it does not consider battery life and has therefore been omitted. For the final assessment, the remaining algorithms—greedy, Rule-based, and the weekly version of LP—have been compared according to the criteria established in the introduction (see Table 5.19). Typically, when comparing algorithms, **runtime** is a key factor. However, in this study, with the given system specifications, the decision variables for the two-year time interval were computed in under two minutes. On average, Greedy, Rule-based, and LP algorithms completed their runs in 8.33 seconds, 1 minute and 1 second, and 1 minute and 3 seconds, respectively. Considering that, in a real-world scenario, only a fraction of the two-year period would need to be computed—meaning that for Greedy and Rule-based, a single iteration for each 15-minute interval would suffice, while LP would require optimisation over just a one-week data window at a time—the runtime of these algorithms would not present a significant difference.

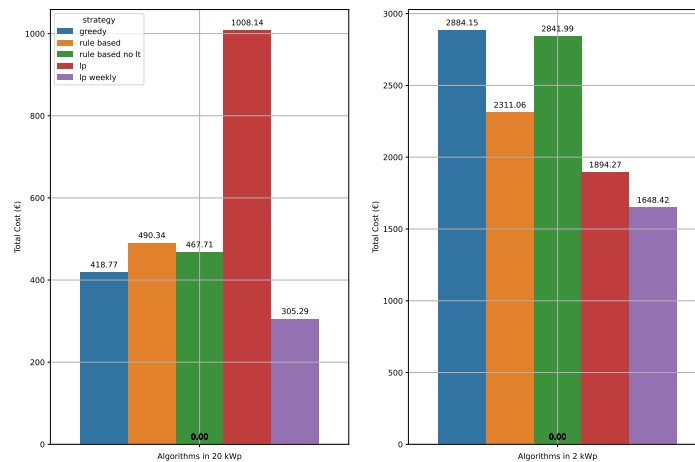


Figure 5.24: Cost comparison of the best and worst test cases

Aspects	Greedy	Rule based	LP (weekly)
Overall electricity cost	1484.5 €	1304 €	851 €
electricity sold/purchased	3415.3/7630.8 kWh	4633/8848.4 kWh	8128.2/12343.7 kWh
Net revenue	-745.2 €	-434.8 €	-128.1€
Complexity	Low	Low	Medium
Forecast requirement	No forecast needed	No forecast needed	Forecast data needed

Table 5.19: Overall comparison (6 kWp)

The measurements in the table are from the medium (6 kWp) test case. In terms of both the primary objective and the additional net revenue, LP significantly outperforms the other algorithms across all cases. However, when factoring in grid dependence, it becomes clear that, in every scenario, including the best and worst cases (see Figure 5.25), LP relies heavily on external energy sources to achieve its optimisation. In general, LP introduces significantly higher import and export overhead compared to the other two algorithms, which could make the other approaches more suitable depending on the specific needs of the community.

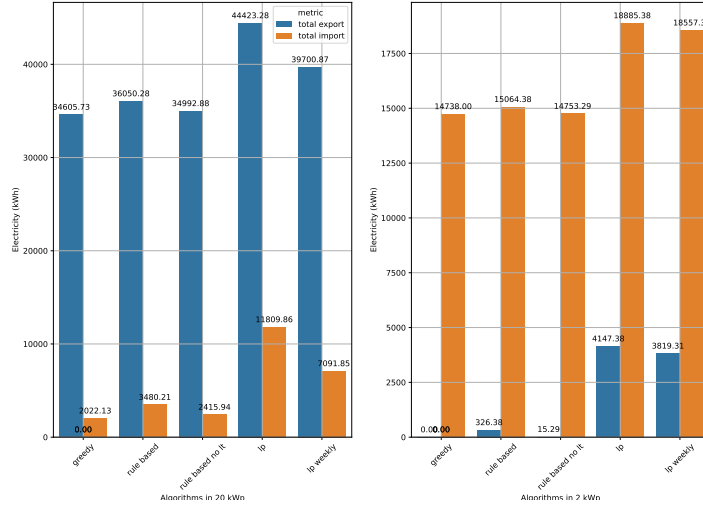


Figure 5.25: Export and import comparison of the best and worst test cases

Regarding the complexity of each algorithm, the first two algorithms (Greedy and Rule-based) can be implemented easily without much difficulty. The same cannot be said for LP. Even when using an open-source solver, the process of modeling the problem, especially in a real-world setting, requires additional error management to account for forecast data deviations. LP's reliance on accurate data is another significant concern. While Greedy and Rule-based algorithms do not require forecasting (with the Rule-based approach using predefined DAM data for determining UT), LP needs accurate forecasts—at least a week's worth—of energy production, consumption, and pricing in order to function effectively. This introduces further development challenges, requiring the implementation of a forecasting module capable of predicting multiple data sources.

6 FUTURE DEVELOPMENT

This project can be improved with further research on battery life, that maps the effective cost to its usage, making it a quantitative cost that can be added to the cost function. With that we can improve the linear programming algorithm used in this thesis work by also considering the battery usage. Furthermore the project can be extended with a forecasting module responsible for predicting the three main datasets: production, consumption and energy price. So we can use the algorithms not only on past, but live data as well. The system should determine data at least a week ahead of time and as precisely as possible. All in all, the future development opportunities are the following:

- Inclusion of battery degradation cost (impact of operating strategy on lifetime)
- Improving the net revenue LP algorithm to preserve battery with the new cost function that includes battery costs
- Developing a forecasting module, that predicts the production, consumption and price data, to test the algorithms on live data

SUMMARY

This study focuses on an optimisation problem identified in the literature: minimising electricity bills for energy communities that can access local energy sources. A test environment was created to mimic the energy consumption characteristics of an educational institution. This environment utilised publicly available time-series data, including a photovoltaic (PV) production profile from Solar Edge, aggregated consumption data from MVM, and energy prices from the Hungarian Power Exchange's Day-ahead Market. A Battery Energy Storage System (BESS) was also incorporated, as it plays a critical role in energy management decisions.

Three optimisation algorithms aimed at reducing energy costs were selected for comparison: Greedy, Rule-based, and Linear Programming (LP). These algorithms make decisions at each time interval about how much energy to draw from or store in the BESS, and how much to import or export. The comparison criteria included overall electricity costs, total imports and exports, complexity (ease of implementation), and additional data dependencies, such as the need for forecast data.

The Greedy and Rule-based algorithms are rule-based designs. The Greedy algorithm focuses on using the battery whenever possible, while the Rule-based algorithm incorporates two price thresholds: Upper Threshold (UT) and Lower Threshold (LT). These thresholds dictate when to import energy or utilise local resources. The Rule-based algorithm draws from the battery only when prices exceed the UT threshold and imports from the grid when prices fall below the LT threshold. Both algorithms make decisions based on the current state parameters of the system.

In contrast, the LP algorithm relies on forecast data. It employs an open-source Linear Programming solver to determine the optimal solution for minimising energy costs over specified time periods. Two versions of LP were analysed: one using daily forecasts and the other using weekly forecasts. Instead of actual forecast data, this study used test data, allowing the algorithms to operate with perfect foresight of energy consumption, production, and price trends. Additional variations of the algorithms were included in the comparison, such as the Rule-based algorithm without the LT threshold and an LP version optimising for net revenue instead of cost, assuming commercial access to the grid.

The cost performance of the algorithms was evaluated across five test cases, with production-to-consumption ratios of 2.63, 1.05, 0.79, 0.52, and 0.26. The comparison between the Greedy, Rule-based, and Rule-based without LT algorithms showed that the no LT version did not provide effective cost savings compared to Greedy. In contrast, the original Rule-based algorithm significantly reduced costs, especially as the production-to-consumption ratio decreased. Greedy outperformed Rule-based only in the highest production scenario.

Comparing the cost performance of all algorithms in 5.2 demonstrated that Linear Programming (LP) emerged as the best algorithm for cost optimisation, but its effectiveness heavily depended on the quality of forecast data. The daily forecast version performed poorly, while

the weekly version consistently yielded the best results in each scenario. Based on 5.3, for meaningful improvements, a forecast interval of at least one week is necessary; longer intervals offered minimal benefits. LP was also tested for optimizing net revenue. Although it showed promise, its heavy reliance on battery usage could shorten battery lifespan, leading to increased costs. Thus, the current form of LP is not suitable for practical implementation.

In conclusion, although the weekly version of LP proved to be the best-performing candidate, it has some notable drawbacks compared to its simpler counterparts, Greedy and Rule-based. Its optimisation process requires significantly higher imports and exports of energy than the other two algorithms, which may not be desirable in scenarios where self-sufficiency or minimising grid imports is a priority.

LP also faces disadvantages in terms of complexity and data dependence. While implementing the model is straightforward, an external solver is needed. Fortunately, open-source solvers are available. In a similar study by [10], the COIN-OR Foundation's [17] solver was used, which was also utilised in this study. On the other hand, forecast data dependence may be challenging. As demonstrated in this study, LP requires accurate forecast data for at least one week ahead, covering energy consumption, production, and prices. This means a forecasting module (or modules) needs to be developed, capable of predicting these time-series datasets with high precision. This would require addressing discrepancies between forecasted and actual data, as well as gathering additional data for reliable forecasting. Based on these considerations, I propose the following recommendations for using the examined algorithms:

- LP: If at least a weekly forecast data of consumption, production and energy price is available, and the community's sole focus is on reducing energy costs.
- Rule-based: The original version with LT and UT should be used. If forecasting is not feasible or if cost optimisation is a secondary objective and self-sufficiency is more important.
- Greedy: In rare cases where production significantly exceeds total consumption.

7 ACRONYMS

LEC	Local Energy Community
PV	Photovoltaic
DSM	Demand side Management
BESS	Battery Energy Storage System
EMS	Energy Management System
ToU	Time of Use
RTP	Real Time Pricing
LT	Lower Threshold
UT	Upper Threshold
PSO	Particle Swarm Optimisation
BPSO	Binary Particle Swarm Optimisation
LP	Linear Programming
MILP	Mixed-Integer Linear Programming
ADMM	Alternating Direction Method of Multipliers
GA	Genetic Algorithms
DSO	distribution system operator
kW	kilowatt
kWp	kilowatt peak
MW	megawatt

8 BIBLIOGRAPHY

- [1] E. Barabino, D. Fioriti, E. Guerrazzi, *et al.*, “Energy Communities: A review on trends, energy system modelling, business models, and optimisation objectives,” *Sustainable Energy, Grids and Networks*, vol. 36, p. 101 187, Dec. 2023, ISSN: 2352-4677. DOI: 10 . 1016 / j . segan . 2023 . 101187. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2352467723001959> (visited on 03/20/2024).
- [2] H. Zhu and K. Ouahada, “Credit-based distributed real-time energy storage sharing management,” *IEEE Access*, vol. 7, pp. 185 821–185 838, 2019, Publisher: IEEE. [Online]. Available: <https://ieeexplore.ieee.org/abstract/document/8938754/> (visited on 04/12/2024).
- [3] A. Cosic, M. Stadler, M. Mansoor, and M. Zellinger, “Mixed-integer linear programming based optimization strategies for renewable energy communities,” *Energy*, vol. 237, p. 121 559, 2021, Publisher: Elsevier. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0360544221018077> (visited on 03/26/2024).
- [4] A. Rochd, A. Benazzouz, I. Ait Abdelmoula, *et al.*, “Design and implementation of an AI-based & IoT-enabled Home Energy Management System: A case study in Benguerir — Morocco,” *Energy Reports, Technologies and Materials for Renewable Energy, Environment and Sustainability*, vol. 7, pp. 699–719, Nov. 2021, ISSN: 2352-4847. DOI: 10 . 1016 / j . egyr . 2021 . 07 . 084. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2352484721005497> (visited on 03/08/2024).
- [5] Smappee, *How is Self-Consumption and Self-Sufficiency calculated ?* en-GB, 2024. [Online]. Available: <https://support.smappee.com/hc/en-gb/articles/360044277371-How-is-Self-Consumption-and-Self-Sufficiency-calculated> (visited on 07/10/2024).
- [6] N. Li, R. Hakvoort, and Z. Lukszo, “Cost allocation in integrated community energy systems—social acceptance,” *Sustainability*, vol. 13, no. 17, p. 9951, 2021, Publisher: MDPI.
- [7] Á. Manso-Burgos, D. Ribó-Pérez, T. Gómez-Navarro, and M. Alcázar-Ortega, “Local energy communities modelling and optimisation considering storage, demand configuration and sharing strategies: A case study in Valencia (Spain),” *Energy Reports*, vol. 8, pp. 10 395–10 408, 2022, Publisher: Elsevier.
- [8] I. F.G. Reis, I. Gonçalves, M. A.R. Lopes, and C. Henggeler Antunes, “Business models for energy communities: A review of key issues and trends,” *Renewable and Sustainable Energy Reviews*, vol. 144, p. 111 013, Jul. 2021, ISSN: 1364-0321. DOI: 10 . 1016 / j . rser . 2021 . 111013. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1364032121003038> (visited on 03/20/2024).

- [9] J. Sousa, J. Lagarto, C. Camus, *et al.*, “Renewable energy communities optimal design supported by an optimization model for investment in PV/wind capacity and renewable electricity sharing,” *Energy*, vol. 283, p. 128 464, 2023, Publisher: Elsevier.
- [10] R. Duvignau, V. Heinisch, L. Göransson, V. Gulisano, and M. Papatriantafylou, “Benefits of small-size communities for continuous cost-optimization in peer-to-peer energy sharing,” *Applied Energy*, vol. 301, p. 117 402, 2021, Publisher: Elsevier. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306261921008023?casa_token=G7Spi4Z21UEAAAAA:f0cnT-KBVfNBv1_2YFTRaTGGYLFwZHWwVfwqDh3NgcIymodMi_13zktOHj-fTq6YL77QF3BIuNX4 (visited on 03/15/2024).
- [11] E. of Encyclopaedia Britannica, *Linear programming*, 2024. [Online]. Available: <https://www.britannica.com/science/linear-programming-mathematics> (visited on 04/16/2024).
- [12] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers, Inc., 2011, vol. 3, pp. 1–122. [Online]. Available: <https://www.nowpublishers.com/article/Details/MAL-016> (visited on 04/16/2024).
- [13] J.-W. Xiao, Y.-B. Yang, S. Cui, and X.-K. Liu, “A new energy storage sharing framework with regard to both storage capacity and power capacity,” *Applied Energy*, vol. 307, p. 118 171, 2022, Publisher: Elsevier. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0306261921014422> (visited on 04/16/2024).
- [14] K. Deb, “An introduction to genetic algorithms,” en, *Sadhana*, vol. 24, no. 4-5, pp. 293–315, Aug. 1999, ISSN: 0256-2499, 0973-7677. DOI: 10.1007/BF02823145. [Online]. Available: <http://link.springer.com/10.1007/BF02823145> (visited on 04/16/2024).
- [15] H. Awad and M. Gül, “Optimisation of community shared solar application in energy efficient communities,” *Sustainable cities and society*, vol. 43, pp. 221–237, 2018, Publisher: Elsevier. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2210670718305821> (visited on 03/25/2024).
- [16] S. H. Kim and Y.-J. Shin, “Optimize the operating range for improving the cycle life of battery energy storage systems under uncertainty by managing the depth of discharge,” *Journal of Energy Storage*, vol. 73, p. 109 144, 2023, Publisher: Elsevier. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2352152X23025422> (visited on 10/04/2024).
- [17] *COIN-OR: Computational Infrastructure for Operations Research*, en-US. [Online]. Available: <https://www.coin-or.org/> (visited on 12/03/2024).
- [18] S. Edge, *Műszerfal*, 2024. [Online]. Available: https://monitoringpublic.solaredge.com/solaredge-web/p/site/public?name=Secutron%20Kft-Smart%20Solar-Az%20intelligencia&locale=hu_HU#/dashboard (visited on 07/11/2024).

- [19] mvm, *Statisztikai elemzéssel készített felhasználói terhelési profilok MVM Hálózat*, 2024. [Online]. Available: <https://mvmhalozat.hu/aram/oldalak/1871> (visited on 07/11/2024).
- [20] HUPX, *Historical data*, en, 2024. [Online]. Available: <https://hupx.hu/en/market-data/dam/historical-data> (visited on 07/11/2024).

9 LIST OF FIGURES

2.1	LEC system with central and individual battery and energy generation [6]	3
2.2	Example for PSO searchspace [4]	12
3.1	Working environment	16
4.1	Greedy algorithm activity diagram	21
4.2	Rule-based algorithm activity diagram	22
4.3	Greedy algorithm activity diagram	23
5.1	Cost, revenue and net revenue of all the strategies with 8 kWp	30
5.2	Import and export of all the strategies with 8 kWp	31
5.3	Cost, revenue and net revenue of all the strategies with 6 kWp	31
5.4	Import and export of all the strategies with 6 kWp	32
5.5	The cumulative energy cost of the strategies over time with 8 kWp	32
5.6	The cumulative net revenue produced by the strategies over time with 8 kWp .	33
5.7	The cumulative energy cost of the strategies over time with 6 kWp	33
5.8	The cumulative net revenue produced by the strategies over time with 6 kWp .	34
5.9	Cost, revenue and net revenue of all the strategies with 20 kWp	35
5.10	SOC in January 2022 with 8 kWp	36
5.11	SOC in July 2022 with 8 kWp	36
5.12	import and export of strategies January 2022	37
5.13	import and export of strategies July 2022	37
5.14	import and export of Rule-based and Greedy strategies	38
5.15	LP Cost Performance Improvement with Increasing Intervals	39
5.16	Total cost of strategies including net revenue LP with 8 kWp	40
5.17	Total export and import of strategies including net revenue LP with 8 kWp . . .	40
5.18	Comparing SOC of net revenue LP to the other strategies January 2022	41
5.19	Comparing SOC of net revenue LP to the other strategies July 2022	41
5.20	The cost performance of Greedy and Rule based algorithms	43
5.21	The total revenue produced by Greedy and Rule based algorithms	43

5.22	Net revenue of Greedy and Rule based algorithms	44
5.23	Performance difference compared to the Greedy algorithm in percent	44
5.24	Cost comparison of the best and worst test cases	45
5.25	Export and import comparison of the best and worst test cases	46