POLITECNICO DI TORINO

Corso di Laurea Magistrale
in Ingegneria Energetica e Nucleare

Tesi di Laurea Magistrale

Evaluation and optimisation of an energy community: an open-source tool

Relatore
Prof. Maurizio REPETTO

Correlatore
Dott. Paolo LAZZERONI

Candidato
Gianmarco LORENTI

Anno accademico 2020/2021

To everyone who has been part of my life in the last five years and affected it, one way or another.

Abstract

This master's thesis deals with the concepts of Collective Self-Consumption and Energy Communities, which represent a way for final customers to share energy that is locally generated by renewable and high efficiency sources. Despite the long history of 'energy sharing' projects in Europe, the introduction of collective self-consumers and Energy Communities in the European and National regulations is quite a revolutionary event that may, to a certain extent, disrupt the traditional way of producing and consuming energy. The idea behind this work is to support the users, which are the main focus of this new and evolving context, with a free and open-source tool to provide quick and easy-to-grasp evaluations of the performances of a configuration where energy is shared among an aggregate of households through collective self-consumption. The tool optimises the operation of a photovoltaic system combined with a storage system (battery) for a number of typical days that are representative for each month of the year. Assessments about the monthly and/or yearly performances are provided, both for fixed-size or variable-size (parametric) analysis of the photovoltaic and of the battery. The evaluations are performed in terms of self-sufficiency and self-consumption indices, which are useful indicators of the energy that is shared over, respectively, the total consumption and the total production of energy. Among the peculiarities of the tool is the generation of the consumption profiles for the aggregate of household customers, using probabilistic methods. In the light of these considerations, starting from a brief overview on the 'energy sharing' projects in Europe, the recently-introduced framework for Energy Communities is presented, both at a European and Italian level. Afterwards, the routine followed in the tool is widely described, with the aim of providing a documentation for the code too. Hence, both the physical equations used and the algorithms implemented are presented. The latter are written in the programming language Python, which is also free and open-source. The tool is then used to simulate one potential configuration, in order to show and discuss the results that it provides. Lastly, the development platform GitHub is briefly introduced, which hosts a repository where the tool is easily and freely available to potential users and contributors.

Contents

List of Figures

List of Tables

Introduction

The opening to collective self-consumption and energy communities, with the Clean Energy for all Europeans Package, marks the acknowledgement of the crucial role that citizens can play in reaching the decarbonisation and renewable generation goals set by the European countries for 2030 and beyond, as well as achieving the renovation of the energy markets all across Europe [1–3]. This acknowledgment stems from a series of evidences given by the long history of community energy (or shared energy) projects in Europe. In the first place, the inclusion of the local communities in the decision-making and the benefits-sharing increases the social acceptance of renewable projects [4]. Also, decentralising the generation allows to exploit local resources and generate value for the community itself [2]. This can be crucial for a further development of renewable infrastructures, in additions to the investments that citizens and communities can realise. Self-consumption of [renewable] energy is indeed becoming a more and more relevant practice in Europe, and the European Commission itself acknowledges that all final customers, paying a special attention to the *vulnerable* ones, should be able to engage in this activity, whether alone or in collective forms [2]. In brief, the engagement of citizens in the energy matter can be a valuable mean to promote changes in the attitude of energy consumption, to boost energy efficiency in buildings and to fight energy poverty [1, 2, 5].

That being so, it clearly appears how formally recognising collective self-consumption and energy communities, and creating a common framework to support their spread and growth were long overdue necessities. Hence, the European Com-

mission has introduced four new figures: (jointly-acting) renewable self-consumers and active customers, which can be seen as a first step of aggregation of final customers; renewable and citizens energy communities, which are instead legal entities whose main goal is to create economic, social and environmental value for the local community rather than making profits. All these represent a way for final customers to join together and self-produce energy which can be either sold, stored, consumed or shared. Thus, they are expected to switch from traditional, 'passive' consumers to active 'prosumers', who are directly involved in the field of energy (and therefore more aware about it). The directives stress how the Member States should enhance the growth and development of these configurations, in which citizens should be able to participate keeping their rights and duties as final customers. As to Italy, some preliminary steps for the transposition of the directives have been made in 2020, allowing final customers to virtually share electricity that is locally produced from renewable energy sources.

In this context, this work aims at providing an open-source tool for the energetic evaluation of the monthly/yearly performances of configurations where electric energy is virtually shared within an aggregate of households, through the use of the public grid. To evaluate the configuration, the operation of a storage system combined with a photovoltaic installation is optimised during a number of typical days which are representative for a whole year. The objective of the optimisation can either be to maximise the energy shared by the configuration or to minimise the interactions with grid. The independent variables are instead the 'instantaneous' production and the consumption from the aggregate of households, which are hence fixed. The former can be easily obtained from freely available online tools, such as PVGIS, while the latter is simulated using probabilistic methods. This is done starting from the instantaneous power demand of the single electric appliances (loads)

found in each household and building step-by-step the aggregate's load profile, in a bottom-up approach. In this way, the tool is able to follow the randomness in the power demand from a small number of households customers, which usually shows relatively high power peaks. The tool, which is implemented in Python, a free and widely-available programming language, requires minimal inputs from the user. Similarly, it provides quick and easy-to-grasp results, which can serve either for a preliminary comparison between various configurations (with different sizes of the photovoltaic system and/or the battery) or for a deeper evaluation of a single configuration.

In the light of the above, the thesis is organised as follows. In Chapter 1, starting from an overview on the 'energy sharing' projects in Europe, the definitions and the provisions about collective self-consumers and Energy Communities introduced in Europe, with the two Directives EU 2018/2001 and EU 2019/944, and in Italy, with the Decree-Law 162/2019 ('Milleproroghe') and other relevant documents, are presented and discussed. In Chapter 2, the routine followed by the tool is widely described, both in terms of the physical equations used and how they are implemented in the code. The chapter is indeed meant to serve as a documentation for the tool. In Chapter 3, a potential energy community consisting of an aggregate of households is simulated, in order to present and discuss the results provided by the tool. Lastly, in Chapter 4, the version control system Git and the online development platform GitHub are briefly introduced, discussing how the tool is made available for the users and for potential contributors to its further development.

Chapter 1.

A framework for Energy Communities

'Shared energy' projects have been going on for decades, especially in Northern-Western Europe countries, such as Germany or Denmark, examples of which being community-owned wind farms or eco-villages which use locally produced biomass to generate heat for the community [1]. Nevertheless, energy communities have never been formally recognised as legal entities at a European level before the introduction of Renewable Energy Communities and Citizens Energy Communities, respectively, in the Directive (EU) 2018/2001 of the European Parliament and of the Council of 11 December 2018 (2018/2001, in the following) and the Directive (EU) 2019/944 of the European Parliament and of the Council of 5 June 2019 (2019/944, in the following), parts of the Clean Energy for all Europeans Package (Clean Energy Package, in the following). The two directives also establish the necessity for member states to create a supportive framework for the emergence and growth of energy communities, confirming the importance of citizens in reaching the sustainability objectives for 2030 and beyond, and recognizing their collective forms as relevant actors in the future energy system [1–3].

The aim of the chapter is to introduce the concepts of self-consumption, especially in its collective form, and 'shared energy' projects. Some of their implications, from a technical, social and environmental point of view are briefly discussed, alongside the framework recently introduced to formally acknowledge these practices in Europe and in Italy. Hence, in the following, building on some evidences about

'shared energy' projects in Europe, discussed in 1.1, the contents of the two European directives EU 2018/2001 and EU 2019/944 are introduced in 1.2. Lastly, the steps made so far to transpose the two directives into the Italian National Regulation, through the Decree-Law 162/2019 and other relevant documents, are presented in 1.3. Technical and regulatory issues that may arise when transposing the directives into National Laws are addressed as well.

1.1 Overview on shared energy projects in Europe

'Shared energy' (or 'community energy') is a wide-ranging and complex topic, with a number of nuances and implications from the most various points of view and discussing it exhaustively would be a difficult task in this context. The objective of what follows is rather to present some relevant aspects that emerged from a brief literature review about community energy projects in Europe, that may help understanding how did it get to the definitions of energy communities and the provisions adopted by the European Commission in their regard. Community energy can be loosely defined as *a series of actions that foster the involvement of citizens and the collective participation in the energy system* [1]. In such perspective, a huge number of community energy initiatives can be identified in Europe, with over 3500 renewable-energy cooperatives, that are the most common type of community energy project [1]. Anyway, it is to be noted that not all these initiatives fall under the two definitions of energy communities provided by the European Commission in the two directives 2018/2001 and 2019/944 [1, 5], that are discussed later on.

Many studies provide a systematic review of the shared energy projects that are currently-running in certain European countries, some of which being: [1], aiming at identifying their activities, organisational forms and drivers; [5], which addresses

technical and regulatory issues and challenges that may arise from the implementation of the provisions introduced with the Clean Energy Package at a national level; and [4] that provides a reading of community energy projects through the concept of 'social innovation'.

1.1.1 Social and environmental implications

The origin of community energy is generally associated with the arise of pro-environmental and anti-nuclear sentiments in the '60s and the oil crisis in the '70s [1, 4]. Anyway, a large development and spreading of community energy projects in Europe occurred when supportive schemes, such as the Feed-in-Tariffs (FiTs) for renewable energy sources, have been introduced [4, 5]. A third phase of community energy's development can be identified in the recession caused by the worldwide financial crisis in 2008, as an expression of the will to democratise the energy system and decentralise the power from the ruling *«big energy companies»*, through the citizens' empowerment [4].

In general, different drivers for the rise and development of community energy projects can be identified. Among these are the will to invest in renewable assets and the possibility of deploying local resources (renewable energy sources) to create benefits for the members of a community, such as the supply of electricity at lower prices, or the sharing of revenues [1, 6]. Also the will to be self-sufficient and more independent from the centralised energy system plays a relevant role, especially when electricity prices are excessively volatile or in case of islands and remote areas that are not reached by the national grid [1, 4]. In any case, *«taking the energy matter in one's hands»* [1] appears to be the *leitmotive* of all these topics.

As previously mentioned, supportive policies determined in the past (and may determine in the next future) the mainstreaming of shared energy projects. As a matter

of fact, while the introduction of FiT schemes by the European governments saw a rapid growth in these initiatives, a reduction of supportive schemes caused a shrinking in their number [1] and, in general, in renewable investments from citizens. In Italy, for example, the photovoltaic market experienced a sever halt after the cuts in FiT schemes in 2013 [7]. On the other hand, a relevant aspect of shared energy projects is that economical benefits, may they be either the sharing of revenues or the savings in the electricity bills, do not necessarily represent the only, nor even the main, reason behind the engagement of citizens in the energy matter. In addition to the ones that can be identified as 'self-regarding' motives, indeed, social and moral norms, such as the environmental concern, the social identification in a community and the interpersonal trust, play a relevant role [6].

In this perspective, energy communities can be seen not only as a technological innovation in the energy system but as a *social innovation* that may disrupt the way the energy system is traditionally seen [4]. They address indeed prominent social issues, which instead might have been overlooked in the past. Among these are the *energy justice*, defined *«as a global energy system that fairly disseminates both the benefits and costs of energy services»* [8], and *energy democracy* [4]. The latter is defined as *«ensuring access to energy for all, producing it without harm to the environment or climate, democratization of the means of energy production and rethinking our attitude to energy consumption»* [9]. Some of these aspects, such as the collective ownership and governance of renewable generation assets and the active participation of citizens in the energy system are recurrent themes in the field of community energy [4, 10].

For instance, shared energy projects introduce a series of benefits, not only for their members but generally for the whole local community [1, 4, 6]. These can be either economical, such as the creation of local jobs and the reduction of outsourcing,

or environmental, such as the reduction of carbon emissions and of fossil fuel usage and the switch to renewable production [1, 4]. Moreover, revenues from the economical activity of energy communities can be re-invested in other projects, further increasing the generation of local value. On the contrary, company-led projects may reduce the 'local' benefits, both because of the profit needed by the company and the lack of control over the project for the community [1]. Researches show in fact how a deep rooting of organisations in the local communities can foster the social acceptance of renewable energy projects. The same holds for the involvement of local citizens in the decision-making process and in the benefits sharing of a project [6].

A relevant aspect is the *locality* feature of both energy communities and renewable energy, which is in fact well-suited to distribute generation. That being so, the emergence and development of energy communities can at the same time help exploiting local resources and ramp up the spread of renewable infrastructures in the next future [1]. In some European countries indeed, a great part of the renewable capacity is owned by individuals or communities. In Germany community energy initiatives can account for almost half of the renewable capacity [1, 6]. Some estimates foresee that *«over 264 million or half of European Union citizens could be producing their own energy by 2050»* [1, 11] and *«about 37% of energy produced by energy citizens could come from collective projects such as cooperatives. Together with small businesses, households and public entities, these groups could own as much as 45% of Europe's renewable generation by 2050»* [1, 12].

Furthermore, energy communities are meant to make citizens switch from passive consumers to prosumers, i.e. producers and consumers, who are actively involved in the energy matter, thus increasing their awareness in the energy-related and climate change-related topics and changing their attitudes towards energy consumption [1].

1.1.2 Activities and technical issues

Currently active community energy projects are involved in a number of activities, ranging from the generation of energy, may it be electricity from solar panels and wind turbines or heat from biomass, to the supply of electricity or the provision of energy services to the members of the community [1]. In some cases, these communities operate an electrical grid or a district heating network, thus working as distribution system operators (DSOs) [1, 5]. Usually, when referring to the *generation*, sharing and self-consuming the energy locally produced among the members of the community are not taken into account, since this is sold to the grid. On the other hand, when referring to the *supply*, it may be either of energy bought from the national grid or of energy produced by the community [1].

While the generation (and sell) of energy from self-owned assets is generally not problematic from a regulatory point of view, some problems may arise in identifying the actual impact on the energy system of the sharing and self-consumption of energy through the use of the public grid [5]. When consuming locally produced energy, the losses in the main energy system decrease, since the transit of energy in the transmission and distribution lines is reduced. Of course, the costs for operating these lines decrease as well. However, the only way to technically and physically ensure that all the whole production is consumed locally would be operating in perfectly-islanded mode (meaning that the community is served by its own grid, separated from the main one) [5]. When the public grid is used to share energy within a community instead, the real savings for the system are to be taken into account [1, 5]. In such cases, virtual schemes represent a viable option [5]. When a virtual scheme is applied, the energy produced is injected into grid (unless it is consumed above the connection point) and the energy consumed is also taken

from the grid; still, a net energy is 'instantaneously' considered, accounting for the difference between the energy produced and the energy consumed. Clearly, the virtual sharing of locally produced energy brings real savings to the grid, in terms of reduced losses, when the production and the consumption *instantaneously* match, or the difference between the two is as small as possible [5]. In most cases, this requires a demand-side management, i.e. the changing in end-user's behaviours in terms of load profiles [5]. Anyway, a similar result can be achieved through the use of a storage system, such as a battery.

Since it is likely that most of the energy communities will still remain connected to the main grid [1], different studies stress the need to take into account the real savings for the energy system and to introduce cost-reflective fees when implementing the directives from the Clean Energy Package into the national laws. This way, it will be ensured that gains for the members of energy communities (in terms of savings on the energy consumed) will not translate into costs for the other customers, thus bringing value to the system as a whole [1, 5].

1.1.3 Organisational forms

While shared energy projects can take various organisational forms, most of the currently active projects are cooperatives. These are strong-social entities, where the one-person-one-vote rule applies. The participation in cooperatives is open and voluntary and they are autonomous and independent. Moreover, they are mainly meant to generate local value rather then profits for their members. Indeed, the revenues are usually reinvested and in any case the distribution of revenues between members is capped [1, 6]. These general principles are also stated by the International Cooperatives Alliance [6, 13]. As such, cooperatives seem to perfectly fit the description of community energy projects that has been pictured until now.

1.1.4 Timeline of past and future events

As previously mentioned, before the introduction of the two European directives 2018/2001 and 2019/944, energy communities were not officially recognised by almost any European country, nor there were specific measures about collective self-consumption. Therefore, citizens who wanted to act collectively in order to produce and sell or share and consume their own energy had to adapt to their National Laws [1]. Yet, shared energy projects thrived in Europe [5], especially in those countries where they could benefit from a more supportive environment. The opening to energy communities with the Clean Energy Package (2016) has represented a turning-point. Indeed, not only energy communities and collective self-consumption are now formally recognised at a European level, but Member States are also supposed to create a supportive framework for their spread and development [2, 3]. The Member States are expected to transpose the provisions adopted by the two directives into their National Laws, also clarifying some aspects that have been intentionally left open to interpretation [5]. Anyway, they should guarantee that the customers' rights are respected and that energy communities *«do not face undue barriers or create undue market distortions»* [5].

The deadlines for the transposition of the directives EU 2018/2001 and EU 2019/944 are, respectively, 30 June 2021 and 31 December 2020 [2, 3]. In Italy some preliminary steps have been made in this regard in 2020[1], with the Decree Law 162/2019 from the Italian Government, the Resolution 318/2020, from the Italian Regulatory Authority for Energy, Networks and Environment (ARERA) and the Decree of 16 September 2020, from the Italian Ministry of Economic Development.

[1] As to the Directive 2019/944, it should be noticed that, even though the deadline for its transposition has already passed, at the moment no measures have been taken in Italy.

Lastly, in December 2020 the Italian Manager of Energy Services (GSE) has issued the technical rules that apply to collective self-consumers and energy communities. The timeline in Figure 1.1 shows how shared energy projects evolved in Europe, together with the steps made so far to introduce a legal framework for energy communities in Europe and in Italy.

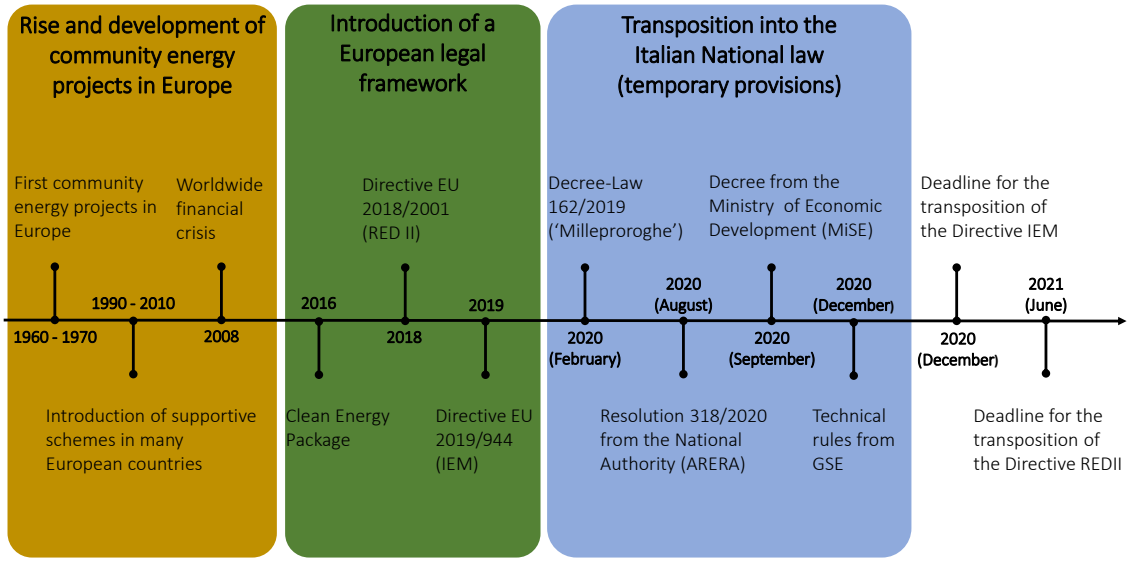


Figure 1.1: *Evolution of community energy projects and of the legal framework for energy communities in Europe and Italy. [Self-processing based on [4, 14]]*

1.2 The framework in Europe

The Clean Energy Package has set the objectives for the European countries in terms of greenhouse gas emissions, renewable production and energy efficiency for 2030 [15]. Citizens are expected to play a relevant role in the realisation of this energetic transition. The European Commission recognises indeed that their involvement in the energy matter can boost the use of renewable energy sources, the diffusion of good practices in the consumption attitudes and the fight of energy

poverty [1, 14]. In this context, the directive 2018/2001 aims at paving the way for a further development of renewable installations, by requiring the Member States to create a supportive framework for the spread of citizens initiatives such as renewable energy communities [5, 14]. The objective of the directive 2019/944 is instead to renew the energy markets all across Europe. Therefore, it aims at creating a *«level-playing field»* for energy communities and citizens, recognising their role as new actors in the energy system [5]. As previously mentioned, the two directives introduce two pairs of figures: (jointly-acting) renewable self-consumers and active customers, renewable and citizens energy communities. Despite sharing many similarities, they show some key differences that stem from the different nature of the two directives [5, 14].

Before moving to a detailed description of these figures, some key aspects about individual and collective self-consumption and energy sharing are introduced. The configurations represented in Figure 1.2 can be useful for this purpose.

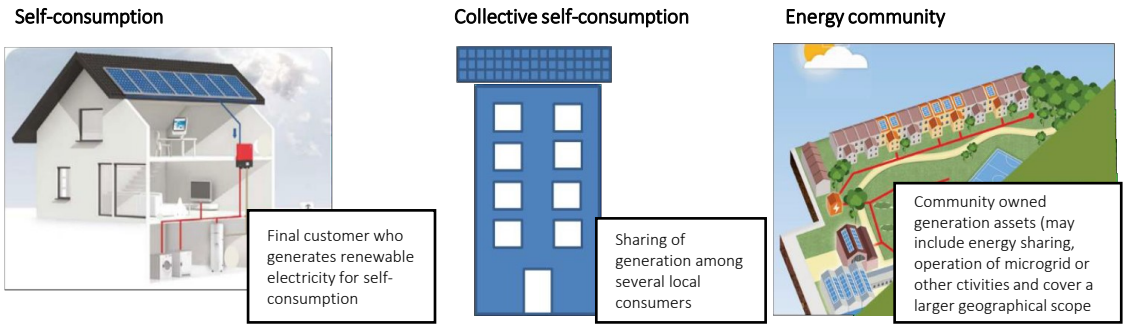


Figure 1.2: *Configurations for self-consumption with progressive levels of aggregation of prosumers. [Source: CEER [5]]*

Individual self-consumption is a well-developed practice across Europe [5]. It is reasonable to assume that the most representative example, in the domestic context, is an household owning a photovoltaic system, which is used to (partially)

fulfill their electricity demand. Of course, in general it can be more complex than this. For instance, the installation may be owned by a third party. In any case, one can consider individual self-consumption whenever energy is produced and consumed locally, regardless of who the producer and the consumer are [16], as long as they are unique subjects (one-to-one configuration). Collective self-consumption, instead, can involve more consumers and one or more producers (one-to-many or many-to-many). They are bounded, however, by some perimeter constraints. For instance, it can be required that the self-generated energy is shared within the same building. Lastly, energy communities involve a multitude of subjects, not necessarily constrained on the perimeter (or, in any case, with a wider geographical scope). Energy is produced from the installations owned by the community itself and it is shared among its member. Energy communities are full-fledged organisations that, in a wider perspective, may engage in a number of energy services (from charging stations for electric vehicles to the operation of a micro-grid) and participate as actual actors in the energy markets.

1.2.1 Renewable self-consumers and active customers

Renewable self-consumers and active customers can be considered as a first level of aggregation, in the transition from traditional, 'passive' consumers to prosumers who join together to produce and share energy for their own consumption [14]. Final customers can become either (jointly-acting) renewable self-consumers or active customers, provided that this is not their primary economical activity[2]. In both cases they are allowed to generate *electricity* within their premises for their own consumption. Only renewable energy sources are allowed for renewable self-consumers,

[2]This does not hold for household customers, in the case of renewable self-consumers.

while there is no limitation on the technology for active customers. In any case, the self-generated can also be stored or sold. It should be noticed that there are no statements about the objective of these configurations, thus meaning that a group of final customers may join together just to obtain a reduction in their energy bills or gain revenues from the energy that is sold to the grid [14]. In addition, the directives state that the production installations can be managed by third parties, without the latter being considered part of the configuration. Anyway, the third party is supposed to be subject to the instructions of the customer [2, 3].

As far as jointly-acting renewable self-consumers are involved, they should be located in the same building or multi-house block. Moreover, according to the directive 2018/2001, they should not be subject to network charges that are not cost-reflective for the energy that is consumed from/fed into the grid. Similarly, they cannot be subject to any fee for the energy that *remains* within their premises[3], provided that not applying such fees does not affect the financial stability of the system [2]. However, the application of fees to the self-generated energy should be done considering the costs of the whole system and ensuring *«that renewable self-consumers contribute in an adequate and balanced way to the overall cost sharing of the system when electricity is fed into the grid»* [2]. As to the renewable energy that is self-generated and fed into the grid, it should be rewarded also considering the value that it brings to the system also on a longer term. It is worth noticing that, in the same directive (Article 7, paragraph 2) it is stated that the renewable energy produced for self-consumption purposes should count for reaching the national objectives in terms of renewable shares [2]. The definitions of renewable

[3]The energy that *«remains within their premises»* can be intended as the physical self-consumption, as discussed later on.

self-consumers and active customers are summarised in Table 1.1, together with the provisions adopted in their regard.

Table 1.1: *Comparison between renewable self-consumers and active customers. [Self-processing based on [2, 3, 14]]*

Type of configuration	**Jointly-acting renewable self-consumers**	**Active customers**
Participation	Final customers, provided that (unless households) it does not represent the main economical activity	Final customers, provided that it does not represent the main economical activity
Physical perimeter	Members should be located in the same building or multi-apartment block	Members and installations should be located within premises with specified boundaries
Energy vector	Electricity only	Electricity only
Energy source	Only renewable	No limitations
Ownership and management of the installations	Third parties can manage and even own the installations but they must be subject to the self-consumers' instructions. They are not considered part of the configuration	Third parties can manage the installations. They are not considered part of the configuration
Activities	Consumption, storage and sell of self-generated electricity, even through peer-to-peer exchanges[a]	Consumption, storage and sell of self-generated electricity, flexibility and energy efficiency schemes

[a] Peer-to-peer (P2P) electricity exchanges are a business model based on the trading of electricity between consumers, producers and prosumers through an interconnected platform, without the need of an intermediary [17].

1.2.2 Renewable and citizens energy communities

Energy communities represent a further step of aggregation of prosumers [14]. They are defined as legal entities, which have the objective of bringing value (social, environmental, economical) to the local community rather than making profits [2, 3]. The participation in energy communities must be open and voluntary and it is to be extended to those customers who have a lower financial availability, such as

«vulnerable and low-income households» [2]. Moreover, they are independent and autonomous bodies, effectively controlled by their members, which can be natural persons, local authorities and enterprises[4]. In the case of renewable energy communities, the members should be located in the *proximity* of the assets owned by the community itself. Members States should allow the customers who are eligible, especially households, to participate in energy communities (or to leave them at any time) without losing their rights and duties as final customers.

The directives state that energy communities should be able to *own* assets to produce energy which can be sold, stored, consumed or shared among their members. When it comes to the sharing of energy within a community, the Member States should guarantee that the DSOs fairly cooperate. In addition, energy communities may as well access all the suitable energy markets without facing any undue barrier but rather taking into account the difficulties they may face due to their generally small size. With respect to the energy that is fed into or taken from the grid, as well as the energy that is shared within energy communities, these should be subject to network charges that are cost-reflective so that they contribute in a fair and balanced way to the overall cost of the system.

As far as renewable energy communities are involved, they can engage in the production of any kind of energy that comes from renewable sources. Citizens energy communities instead can engage in the production of electricity only, with no limitations on the source. They may also engage in activities regarding flexibility, energy efficiency, recharge of electric vehicles and other energy services. As to citizens energy communities, the Member States are given the possibility to choose if

[4]For private undertakings, the participation in a *renewable* energy community should not represent the main financial activity.

they can own, lease and/or operate their own distribution network, in which case they would be subject to the same rights and obligations as any other DSO. The definitions of renewable and citizens energy communities and the main provisions adopted in the two directive are summarised in Table 1.2.

Table 1.2: *Comparison between renewable and citizens energy communities. [Self-processing based on [2, 3, 14]]*

Type of configuration	**Renewable energy communities**	**Citizens energy communities**
Participation	Natural persons, local authorities, medium and small enterprises, provided that it does not represent the main economical activity.	Natural persons, local authorities, small enterprises
Physical perimeter	Members should be located in the proximity of the installations owned by the community	No limitations
Energy vector	No limitations	Electricity only
Energy source	Only renewable	No limitations
Ownership and management of the installation	Not explicitly stated in [2]. Anyway in the definition, *«projects that are owned and developed by that legal entity [the energy community]»* are mentioned [2]. In [14] this is interpreted as *«at full disposal of the community»*.	Not explicitly stated in [2]. Anyway in [3] *«production units owned by the community»* are mentioned . In [14] this is interpreted as *«at full disposal of the community»*.
Activities	Production, storage, consumption and sell of energy. Share of the self-produced energy within the community. Access to all relevant energy markets. Supply and aggregation.	Production, storage, consumption and sell of electricity. Share of the self-produced electricity within the community. Access to all relevant energy markets. Supply and aggregation, flexibility, energy efficiency and electric vehicle recharge services. If allowed by the Member States, operation of distribution networks.

In Italy, the possibility of operating a private distribution grid for energy communities has been strongly discouraged by ARERA. According to the Authority, indeed, efficiently using the existing grid is generally more advantageous than build-

ing new lines, especially in residential contexts [16]. However, there are also other degrees of freedom left to the Member States in the transposition of the two directives into their National Laws. As to the constraint on the perimeter for renewable energy communities, for instance, the concept of 'proximity' is to be properly specified [14]. One possibility is to require members of a community to be located below the same medium-low voltage substation. Anyway, some relevant issues may arise from this choice, which are discussed later on. Furthermore, while the role of third parties in the management of the installations has been clearly stated in the case of renewable self-consumers and active citizen, the Member States should specify how the 'ownership' of the project/installation is to be interpreted in the case of energy communities, as well as the role that third parties can play [14]. Lastly, each Member State is left the duty to define the cost structure to be applied to each configuration, such as network and general system fees, as well as any incentive scheme for the renewable energy that is generated and consumed, shared or sold to the grid.

1.3 The framework in Italy

Individual self-consumption is a long-standing practice in Italy. The Italian Manager of Energy Services (GSE) has indeed estimated that the self-consumed energy from photovoltaic installations in 2018 was larger than 5 TWh, i.e. the 22.7% of the total photovoltaic production [18]. Other estimates report a total self-consumption of around 28 TWh in the same year (slighlty less than 10% of the total electricity consumption), with around 20% of renewable share [16]. Nonetheless, the current framework for self-consumption is the result of a fragmentary and non-linear path [16] and the National Authority (ARERA) has stressed the need to strongly revise and simplify it. A series of private configurations have been indeed legally

recognised over time in order to adapt to the technological changes and to fill regulatory gaps [14]. Apart from few exceptions related to long-established cooperatives, all these configurations can only involve one producer and one consumer (not necessarily coincident). Moreover, a direct connection is needed between the production installation and the consumption unit [14]. On the contrary, the introduction of collective self-consumption and energy communities paves the way for one-to-many or many-to-many configurations [19], where no private connection is required since virtual schemes are applied, as discussed in the following.

1.3.1 Renewable self-consumers and energy communities

As a first step to the transposition of the directive 2018/2001, the Decree-Law 162 of 30 December 2019 (converted into Law n. 8 on 28 February 2020, D.L. 162/2019 in the following) has opened up to collective self-consumption and energy communities in Italy. Anyway, the provisions of the decree have a temporary nature since it *de facto* anticipates the transposition of the two directives, with the objective of starting an experimental phase and to provide evidences for the complete transposition [14, 20]. The decree introduces the figures of jointly-acting renewable self-consumers and renewable energy communities, without adding any change to the definitions provided by the directive 2018/2001 [14].

Hence, final customers can become jointly-acting renewable self-consumers, and natural persons, local authorities and small and medium enterprises can participate in renewable energy communities. In both cases this cannot represent the main commercial or professional activity (unless households, in the case of renewable self-consumers) [20]. Low-income families must be guaranteed the right to participate in energy communities, complying with the objective of the directive 2018/2001 about fighting against energy poverty.

The points of connection with the public grid, or points of delivery (PODs), of the installation(s) and of the members of an energy community must be located on the same low-voltage line, below the same medium/low-voltage substation. In the case of renewable self-consumers instead, the perimeter is identified as the same building or multi-house block. As to the organisational form that these configurations may assume, the decree states that the members must constitute a *«legal entity»* [20] without specifying any particular form. In both cases, final customers can engage in the production, consumption, storage, share and sell of *electricity* from renewable energy source. However, the emphasis is on the self-generation of energy for fulfilling the local demand (*«the subjects participating* [in the configuration] *can produce energy for their own consumption»* [20]). The decree states that the energy must be shared through the use of the existing distribution grid, either instantaneously or through a storage system. Hence, the possibility for energy communities to operate their own distribution line is clearly denied by the Italian Government, as recommended by ARERA. In other words, a virtual scheme applies [14]. The *shared energy* is determined as *«the minimum, in each hour, between the energy that is produced and fed into the grid from the renewable installations and the energy that is taken from the grid by the members of the configuration»* [20].

The decree states as well that the total size of the installations must be smaller than 200 kW. In compliance with the transitory nature of the provisions, they must entry into service within sixty days from the complete transposition of the directive 2018/2001. The characteristics of the two configurations introduced in the Decree-Law 162/2019 are summarised and compared in Table 1.3.

Jointly-acting renewable self-consumers and members of energy communities maintain their rights and duties as final customers (such as the choice of the energy supplier) and can recede from the configuration at any time. As to the internal

Table 1.3: *Comparison between renewable self-consumers and energy communities, according to the Decree 162/2019. [Self-processing based on [14, 20]]*

Type of configuration	**Jointly-acting renewable self-consumers**	**Renewable energy communities**
Participation	Final customers, provided that (unless households) it does not represent their main economical activity	Natural persons, local authorities, small and medium enterprises, provided that it does not represent their main economical activity
Physical perimeter	Members must be located in the same building or multi-apartment block	Members' PODs must belong to the same low-voltage line
Energy vector	Electricity only	
Energy source	Renewable only	
Installations	The total size of the installations must be smaller than 200 kW and they must entry into service after the 1 March 2020 and within 60 days after the transposition of the directive 2018/2001	
Activities	Storage, consumption and sell of self-generated electricity, share of electricity among members of the configuration (following a virtual approach)	

economical regulation of the sharing of energy, the decree just states that *«they* [the final customers participating in one of the configurations] *regulate their relations through private-law agreements»* [20]. Moreover, they are expected to identify a person (*representative*) who is responsible for the sharing of the energy among the members and who can manage the payments and payouts to and from the energy sellers and the GSE.

Lastly, the decree states whether some tariffs can be applied or not to the *shared energy* and how it is supposed to be incentivised:

- The *shared energy* is subject to the general system fees;
- The Authority is supposed to *«determine, even through a flat-rate approach, the value of the components of the energy tariff that cannot be technically applied to the shared energy, since it is instantaneously self-consumed on*

the same low-voltage line and is therefore comparable to a physical self-consumption» [20];

- The Ministry of Economical Development (MiSE) is supposed to determine the incentives aimed at recognizing the value of the renewable energy that is instantaneously self-consumed and/or stored.

Again, it is to be noted how the decree focuses on the self-consumption of the renewable energy that is produced, rather than the production of renewable energy itself, by awarding the amount of energy that is shared.

In its definition, the decree states that the *shared energy* can be considered as physically self-consumed. Anyway, in a virtual scheme, unlike a physical one, there is no direct (private) connection between the production installation(s) and the consumption unit(s), since energy is shared through the public grid. All the electricity production, indeed, is fed into the grid[5]. Similarly, the members of the configuration purchase energy from the grid to fulfill their whole electricity demand, therefore paying their bills as usual. Yet, the energy that is fed into the grid is considered as shared, when it matches the consumption (instantaneously, or through a storage mean), therefore fully comparable to a physical self-consumption. While this may seem odd, it is important to remind that, for a given electric line, the electricity follows the path with a lower impedance (in this case, the shortest path). In other words, the electricity that is fed into the grid is *effectively* consumed locally. As a matter of fact, the application of a virtual scheme over a physical scheme is a regulatory issue rather than a technical one [19]. The components of the energy

[5]Actually, a part of the production may still be consumed before the installation's connection point, i.e. physically self-consumed

tariff that are not applicable to the *shared energy* are refunded during each billing period by the GSE, which is also supposed to hand out the incentives.

1.3.2 Technical and economical regulation

The *Memoria* 94/2019/I/COM [16], from ARERA, presents a number of relevant aspects about self-consumption, which can help understanding the provisions adopted for the economic regulation of the energy that is virtually shared by renewable self-consumers and energy communities. In the document, the Authority emphasises how term 'self-consumption' identifies the practice of consuming locally-produced energy, either instantaneously or through a storage system. As such, self consumption has a series of effects on the energy system, which do not depend on the source of the self-consumed energy, whether renewable or not, and regardless of the producer and the consumer, who do not need to coincide [16]. For this reason, the benefits (as well as the distortive effects) that are to be ascribed to self-consumption in its general form should be taken into account separately from the ones that a particular form of self-consumption, such as renewable, can bring to the system [16]. The Authority had proposed indeed to extend the definition of renewable self-consumers in order not to discriminate between the energy source and the 'commercial structure'. According to this definition[6], self-consumers would have been the wider group of *«consumers who produce and consume energy in the same* [geographical] *site, even when the latter is produced by a third party»*, of which renewable self-consumers would hence have been a subset.

[6]It is to be noted that, at least in the context of the transitory measures introduced by the Decree-Law 162/2019, this definition has not been embraced.

The aforementioned effects that self-consumption induces on the energy system can be related to the following fields:

- Transmission and distribution lines losses;
- Use of the connection points;
- Development of new lines;
- Dispatching.

On the one hand, the electricity flows on the transmission and distribution lines, and consequently the energy losses, (as well as the use of the lines) scale down as the *quantity* of locally-produced and consumed energy increases. On the other hand, to optimise the use of the connection points, the users should require smaller *maximum* ('contractual') powers. The same holds for the development of new lines, that may potentially require lower capital costs, if they are sized to carry a smaller maximum power. As to the dispatching costs, the Authority points out that Terna, the Italian Transmission System Operator (TSO), still needs to procure an adequate backup-capacity in order to meet the demand from self-consumers when their installations are not producing energy. From this point of view, the dispatching costs may even increase, since they scale up with the *volatility* of the energy source [16].

According to these considerations, the Authority argues that the exemption from the variables components (i.e. related to the quantity of energy that is purchased) of the transmission and distribution costs, as well as the costs related to the conventional losses on the lines, represents a just compensation for the self-consumed energy. On the other hand, the general system fees, despite being related to the energy system, do not scale up or down with the quantity of energy that transits in it [16]. There is hence no reason, according to the Authority, to exempt

self-consumers from contributing to them, from the perspective of recognising the beneficial effects of self-consumption. Nonetheless, the directive 2018/2001 states that renewable self-consumers shall not be subject to any fee, including network and system costs, with respect to the energy that is physically self-consumed (*«the energy that remains within their premises»*). It should be noticed, however, that this is already laid down in the regulations for individual self-consumers, since they are charged for the energy that is purchased from the grid only [16]. For the reasons discussed above, such exemption overestimates the actual value of the self-consumed energy and represents an implicit incentive that can be estimated in 1.4 billions of euro each year [16].

Implicit incentives of this kind should be avoided for two reasons. On the one hand, the general system fees still have to be covered, therefore the savings for self-consumers turn into costs for other customers (often the most vulnerable ones, that are not ready, financially, to embrace the same opportunity). On the other hand, implicit incentives are not reliable for the ones who benefit from them, since they may be subject to changes over time, nor can they be properly calibrated on the objective that is pursued, whether it is enhancing the self-consumption of energy or the deployment of renewable energy sources (in case of renewable self-consumers) [16]. Lastly, the Authority points out how, according to the directive 2018/2001, the objective of renewable energy communities is not to maximise self-consumption but rather to bring economical, environmental and social value to their members and to the local communities. Hence, a different 'tariff treatment' should be applied [16].

In the light of the previous considerations, the provisions adopted by the Authority in the Resolution 318/2020/R/eel with regard to collective self-consumers and energy communities can now be discussed. First, the resolution defines the cost

components that are not applicable to the *shared energy* (which are summarised in Table 1.4), considering the effective benefits that the instantaneous consumption of locally-generated energy can bring to the system. These include, for both configurations, the variable cost components related to the use of the transmission and distribution lines. As far as renewable self-consumers are involved, the Authority states that the components related to the energy losses on the same voltage-level line cannot be applied as well. This does not hold for energy communities, since the public distribution line is still used to share energy [21].

Table 1.4: *Cost components that are applicable to the shared energy, according to the Resolution 318/2020. [Self-processing based on [14, 21]]*

Type of configuration	**Jointly-acting renewable self-consumers**	**Renewable energy communities**
Transmission and distribution components	×	×
Distribution losses	×	✓
General system fees	✓	✓

Table 1.5: *Unit prices of some cost components for electricity in Italy. [Source: GME, ARERA [22, 23]]*

Cost component	**Value (c€/kWh)**	**Year**
$TRAS_E$ [a]	0.794	2021
BTAU [b]	0.062	2021
PUN [c]	5.2	2019

[a] Variable component of the transmission costs for low voltage users; [b] Variable component of the distribution costs for other low-voltage users; [c] National Single Price, average between the zonal prices in 2019. The value in 2020 is significantly lower (around 3.9 c€/kWh) mostly due to the Covid pandemic.

In Table 1.5, the unit prices of the previously mentioned cost components for electricity in Italy are shown. Using these values and considering that the coefficients

for the avoided distribution losses on the medium-voltage and low-voltage lines (that are to be multiplied by the unit zonal price) are equal to, respectively, 1.2 % and 2.6 %, the unit compensation for the *shared energy* can be estimated in:

- Around 1.00 c€/kWh for renewable self-consumers;
- Around 0.86 c€/kWh for renewable energy communities.

Furthermore, the resolution clarifies some relevant aspects that were not comprehensively explained in the D.L. 162/2019, concerning the topics that are here briefly highlighted:

- Perimeter of the configurations;
- *Shared energy*;
- Size and ownership of the installations.

As to the former, while belonging to the same building or multi-house block remains a necessary condition for jointly-acting renewable self-consumers, no voltage level is specified for the line. At the same time, the resolution confirms that the PODs of all the members and the installations of energy communities are to be on the same low-voltage line. Anyway, the constraint on the perimeter must be valid only at the establishment of the configuration, allowing for future modifications caused by any potential technical need from the DSO.

As far as renewable self-consumers are involved, the Authority states that the energy consumed by customers who are located in the same perimeter as the configuration but that are not part of the latter can be taken into account in the computation of the *shared energy*, if they agree to. The same does not hold for energy communities. The Authority relaxes the constraint on the total size of the

installations, specifying that the 200 kW limit is to be intended on each single installation. Furthermore, new sections of existing installations can be considered part of the configuration, provided that the energy that they produce can be measured separately. The issue related to the ownership of the installations is somehow resolved for both renewable self-consumers and energy communities. For the former, the Authority states that the installations, including eventually any storage system, can be owned and managed by a third party provided that the latter is always subject to the instructions of the final customers. While this holds also for energy communities, in this case a distinction is made between the *ownership* and the *possession* of the installations. The community can hold the *possession* of the installations without actually *owning* them (the *ownership* can belong to a third party who can also manage the installation), provided that such distinction cannot, under any circumstances, be used to get past the objective of the community [21].

The resolution defines as well the figure of the *producer*, that is any natural or legal person who produces energy, regardless of the actual ownership of the installation. The definition of group of renewable self-consumers is therefore broadened to a group of final-customers *and/or producers* who are located in the same building or multi-house block (as usual, provided that this is not their main commercial or professional activity) [16]. For both renewable self-consumers and energy communities, the installations can be owned and managed by a producer that is not part of the configuration, on the conditions mentioned above.

Before moving to the incentive structure, it is worth discussing some key issues concerning the limitation on the perimeter of energy communities. First, it should be noted that, while the DSOs hold the information about what line a POD belongs to, the approval of the POD's owner is necessary in order to share this information with third parties. Therefore, theoretically, one cannot know in advance which users

to involve in an energy community project [14]. For this reason, it has been proposed to relax this constraint in order to include all customers which share the same postal code as potential members of an energy community [14]. The solution adopted by the Authority is instead that the DSOs foresee online tools where customers can easily check which PODs belong to the same low-voltage line [21]. Anyway, while this may solve part of the problem, it is to be noted that when the size of an installation is larger than 100 kW, it is faculty of the DSOs to decide whether the connection may be on a low-voltage or medium-voltage line. Hence, there are no advance guarantees that energy communities may own installations whose size is larger than 100 kW [14]. Furthermore, medium enterprises are usually connected to medium-voltage lines [14], therefore such constraint on the perimeter may prevent them the possibility to participate in energy communities.

1.3.3 Structure of the incentives

The Italian Ministry of Economic Development (MiSE) has issued the Decree of 16 September 2020 [24], aimed at identifying the incentives for the energy that is self-generated and instantaneously consumed by renewable energy communities and renewable self-consumers. These incentives are to be recognised for a time period that reflects the lifetime of the installations, which is assumed as 20 years, and are equal to:

- 100 €/MWh for jointly-acting renewable self-consumers;
- 110 €/MWh for renewable energy communities.

Of course, this is to be summed to the compensation for the cost components that are not applicable to the *shared energy*, further increasing its economical value.

Anyway, it should be noticed that the latter is not a form of incentive, since there is *effectively* no reason to apply these cost components to the *shared energy*. The incentive for the *shared energy*, on the other hand, is an *explicit* form of awarding the practice of collective self-consumption. This is also meant to replace the other supportive scheme for the self-consumption of renewable energy, namely the net-billing scheme *"scambio sul posto"*, which of course cannot be accessed by the configurations who access the other incentive. The decree states as well that the energy that is self-generated and fed into the grid is to be considered at the disposal of the *representative* of the configuration, who can decide to cede it to the GSE, eventually through the dedicated-withdrawal system (*"ritiro dedicato"*).

Furthermore, accessing the incentive excludes the members of a configuration from the tax deduction scheme *"Superbonus 2020"*. The latter foresees the compensation of 110 % of the expenditures on photovoltaic installations up to 20 kW and on storage systems, implemented in the context of other energy efficiency or seismic risk prevention measures between 1 July 2020 and 31 December 2021. Members of both configurations can access the scheme and still benefit from the compensation of the energy cost components that are not applicable to the *shared energy* (but cannot access the incentive). Anyway, the self-generated energy that is fed into the grid is automatically ceded to the GSE.

Otherwise, they can access a different tax deduction scheme, *"Bonus casa"*, that foresees the compensation of 50 % of the gross expenditure, for the whole set of installations or for the remaining installations from 20 kW to 200 kW (in case they access the *Superbonus*), provided that the energy produced by the latter and the one produced by the installations benefiting from the *Superbonus* can be measured separately. In this case they can still access the incentives, of course with respect to the installations that do not benefit from the *Superbonus*. The two forms of

tax deduction schemes, applied to the case of renewable self-consumers and energy communities, are summarised in Table 1.6.

Table 1.6: *Tax deduction schemes available for energy communities and jointly-acting renewable self-consumers. [Self-processing based on [14, 24]]*

Tax deduction scheme	***Superbonus 2020***	***Bonus casa***
Compensation	110 % of the expenditures on photovoltaic installations up to 20 kW and storage systems	50 % of the gross expenditures on the installations from 20 to 200 kW or, alternatively, on the whole set of installations
Incentives on the *shared energy*	×	✓
Compensation for the cost components not applicable to the *shared energy*	✓	✓
Other incentives	×	×
Self-generated energy fed into the grid	dedicated-withdrawal system	at disposal of the *representative* of the configuration

Ultimately, the total value recognised to the *shared energy* can be estimated as roughly 110 €/MWh for renewable self-consumers and 118 €/MWh for renewable energy communities. The value of the energy that is produced and fed into the grid, instead, can be roughly estimated as 50 €/MWh. If the two quantities coincide (i.e. the whole production is shared), the total value of the *shared energy* is, averagely, around $160 \div 170$ €/MWh, which is very interesting if compared to the price of electricity for households, usually assumed as $170 \div 180$ €/MWh[7].

[7]ARERA [23] has estimated an average price of 16.6 €/kWh for an Italian household whose yearly consumption is 2700 kWh/year, for the first semester of 2021.

Chapter 2.

An open-source tool

According to the provisions introduced in the Italian National Laws with the Decree-Law 162/2019 (and successive documents), final customers can join together to become jointly-acting renewable self-consumers or participate in renewable energy communities. In this context, an open-source tool has been created to provide quick and easy-to-grasp evaluations of the performances of a configuration where locally-produced renewable electricity is virtually shared among an aggregate of households, through collective self-consumption. It is assumed that the renewable installation is coupled with a storage system, namely a battery. Anyway, base-case scenarios without the battery can also be simulated. The objective of this chapter is to describe the tool in detail, thus providing a documentation for the users and potential collaborators who might be interested in a deeper insight on its functioning.

The tool, which is composed of a number of modules written in `Python` language, requires few inputs from the user. Among these are the CSV (Comma-Separated-Values) files containing the specifications about the battery and the production profiles of the renewable installation, which is assumed to be a photovoltaic system. Furthermore, when the code is run, the user is supposed to enter some information about the aggregate and the type of simulation that is to be performed, i.e. the different sizes that are to be explored, both for the photovoltaic, in terms of peak power (kW_p), and the battery, in terms of nominal capacity (kWh).

The data about the aggregate are used to simulate the electricity consumption profiles needed for the subsequent calculations. For this task, a routine has been created, which uses probabilistic methods to build the aggregate's consumption profiles, starting from the power demands of the single electric appliances found in each household. Afterwards, each combination between the sizes of the photovoltaic installation and of the battery is evaluated in terms of yearly performances. To do this, a number of typical days are considered, which are supposed to be representative for the whole year. The power flows realised in the configuration are optimised during each typical day, thus defining the battery's usage strategy. The optimisation objective can be maximising the shared energy or, alternatively, minimising the interactions with the grid. If just one configuration is to be evaluated (fixed-size analysis for both the photovoltaic system and the battery), detailed results about the optimised power flows are provided, together with monthly results about the self-consumption, the self-sufficiency and the shared energy. If more configurations are to be simulated (parametric analysis), the latter are provided on a yearly basis, in order to quickly compare the different configurations' performances.

Being the tool open-source, the code has been written foreseeing future developments from other collaborators. For this reason, easy-to-understand names have been assigned to the variables in `Python`. The drawback is that these names can be quite long. Considering that both the physical/mathematical equations and the algorithms used to implement them in the code are described in the following, a shorter notation is used in the former for the sake of readability. Hence, the same quantity may be addressed using two different designations. Whenever this happens, it is clearly pointed out. Moreover, bold font is specifically used to refer to the **Python variables**, while math font is used for *physical quantities*.

2.1 Model of the system

In compliance with the provisions adopted in the Decree-Law 162/2019 and in the Resolution 318/2020, a virtual scheme is considered in the tool to model the configuration. In Figure 2.1 a physical scheme for collective self-consumption in a building is shown, where the production installation and the consumption units are directly connected to each other. In the virtual scheme represented in Figure 2.2, instead, the electric energy is shared/self-consumed through the public grid. As previously mentioned, this means that the whole production from the renewable installation is fed into the grid and the whole demand from the households (consumption) is fulfilled purchasing energy from the grid. However, it should be recalled that this distinction is only a regulatory issue. From a technical point of view, in fact, the virtual sharing of electricity is effectively comparable to a physical self-consumption. This is ensured by the proximity of the consumption units to the production installation and the fact that, for a given topology of the line, the electricity takes the path characterised by the lower impedance.

It should be noticed that in a physical scheme such as the one represented in Figure 2.1, the households would result as 'hidden customers', due to the presence of only one connection point for the whole configuration. According to the current regulation in fact, apart from few exceptions, each consumption unit (in this case, each household) is supposed to be connected to the public grid through its own POD [25]. Moreover, this particular setup would undermine the final customers' right to choose their own energy supplier, since only one contract would be stipulated for the whole configuration. [25]. A virtual scheme, on the contrary, allows the customers to keep their own POD (therefore the possibility to choose their own supplier) and to recede from the configuration at any time without requiring any

physical operation. Similarly, no physical work is required when the configuration is created. Clearly, this aspect, together with the possibility of preserving the final customers' rights, played a relevant role in the adoption of virtual schemes when the Italian framework for energy communities and collective self-consumption has been defined.

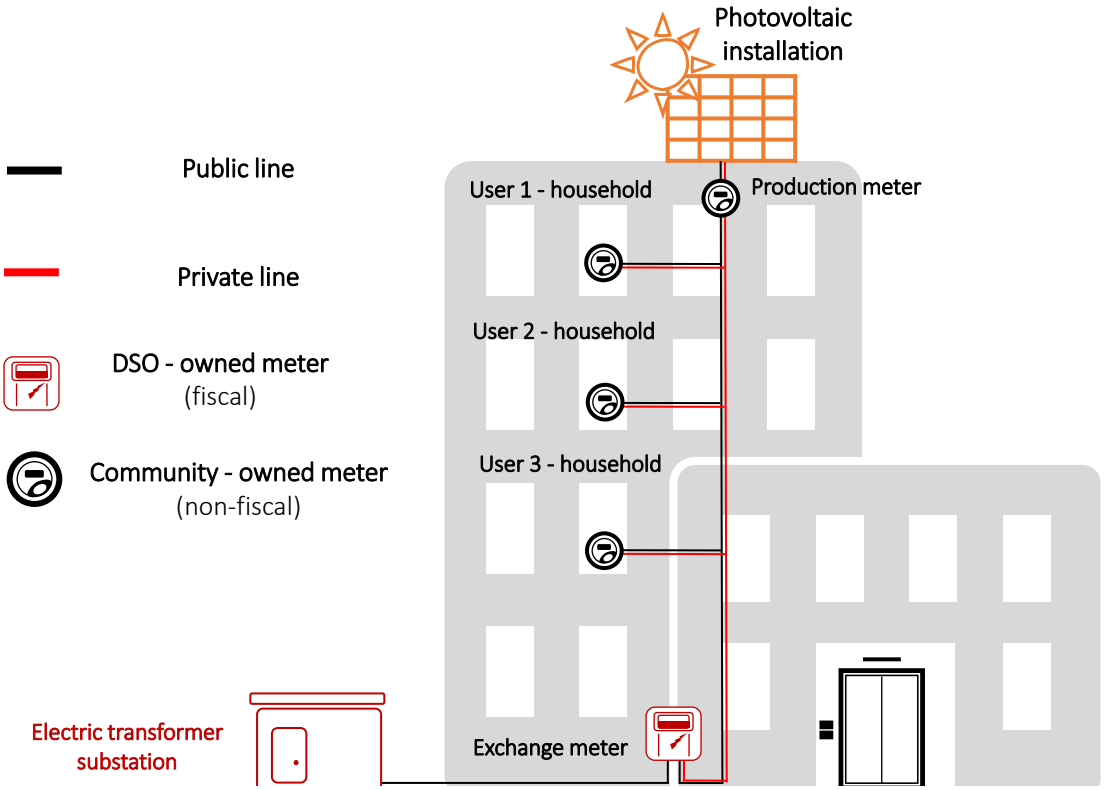


Figure 2.1: *Physical scheme for collective self-consumption in an apartment block. [Adapted from RSE [25]]*

According to the Decree-Law 162/2019, the energy that is virtually shared is defined as the hourly minimum value between the total energy that is fed into and the total energy that is taken from the grid, instantaneously or through a storage system. Anyway, a part of the production could still be physically self-consumed,

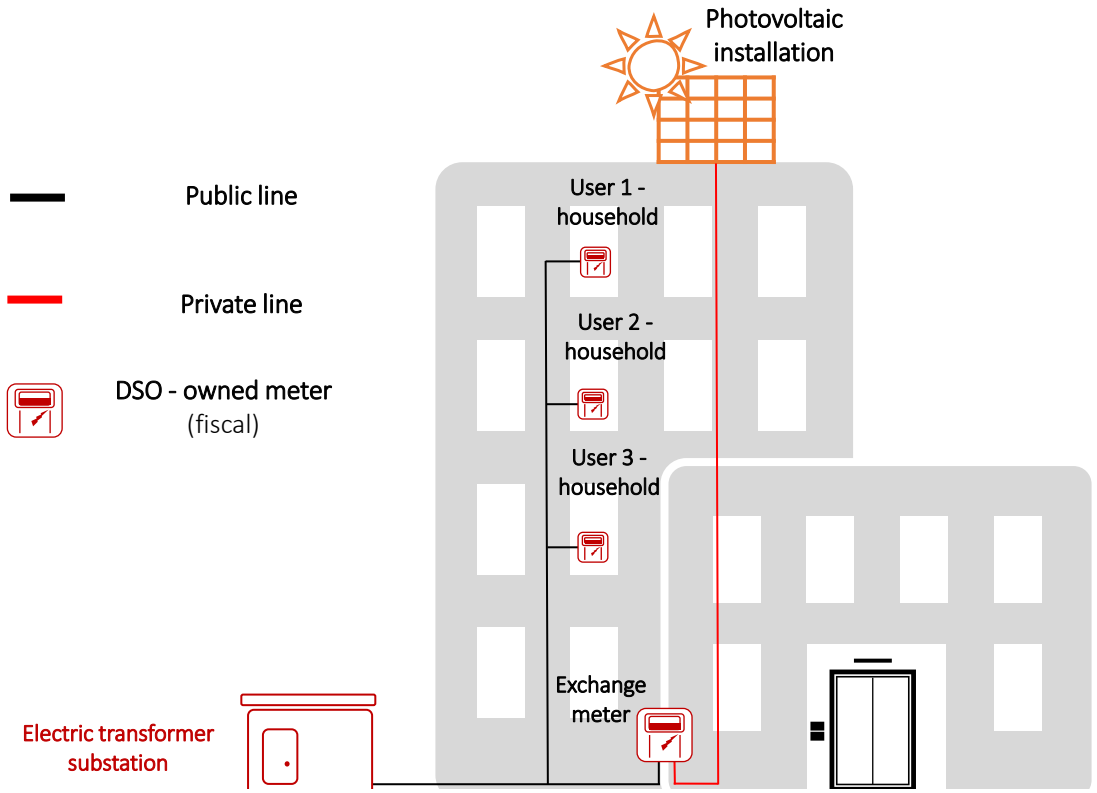


Figure 2.2: *Virtual scheme for collective self-consumption in an apartment block. [Adapted from RSE [25]]*

for example to power some shared services, such as an elevator. In this case, the injections into the grid are to be reduced by the energy that is directly consumed, which cannot count in the evaluation of the shared energy since it is not subject to any fee. In the model implemented in the tool, however, no physical self-consumption is considered mainly due to the fact that the routine that simulates the consumption profiles works for households only. Hence, the whole production from the renewable installation counts in the evaluation of the shared energy.

While the tool works for any configuration where electricity is virtually shared, in the following it is assumed to deal with energy communities of households rather than jointly-acting self-consumers who are located in the same building. The scheme

shown in Figure 2.3 is well-suited to represent the model of energy community that is implemented in the procedure, where the renewable installation is coupled with a storage system. It can be noticed that the latter is connected to the public grid only (not to the installation nor the consumption units), therefore energy is also stored virtually. The configurations allowed by the GSE in the technical rules for the access to the valorisation and incentive of the shared energy [26] can be divided into two macro-groups, depending on the location of the storage system, which can either be on the 'production side' or 'post-production side'. The system modelled in the tool falls under the second category, which has been chosen due to the possibility of considering more renewable installations without having to consider a storage system for each one of them.

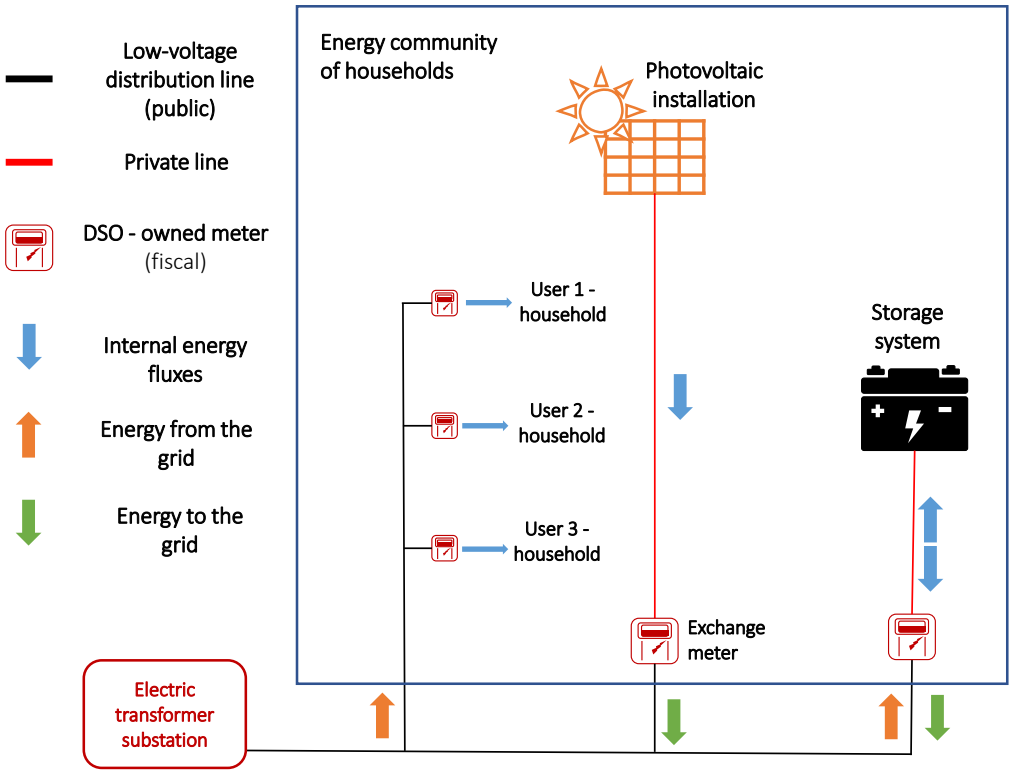


Figure 2.3: *Virtual sharing of electricity in an energy community of households with storage system. [Adapted from RSE [25]]*

2.1.1 Renewable installation

A photovoltaic generator is represented as the renewable installation of the configuration in Figure 2.3. Anyway, according to Decree-Law 162/2019, any kind of system that generates electricity from renewable sources, such as micro-hydro or small-scale wind turbines, would be suitable. Nevertheless, photovoltaic appears to be the most-suited and easiest-to-implement solution in this case, considering both the residential context and the constraints on the perimeter of the configuration [14]. Moreover, despite the market slowdown experienced after the cut of FiT schemes, it is still the most employed technology for self-consumption in the residential context. As a matter of fact, the growth in the number of installations over the last years was mainly due to small-size installations realised in the domestic sector [27].

A detailed description of the photovoltaic system is out of the scope of this work. In the procedure, indeed, only the unit hourly production profiles from the renewable installation during each typical day are needed, whatever the technology is. For a photovoltaic generator, they can be easily obtained from the JRC's (Joint Reasearch Center) online tool PVGIS (PhotoVoltaic Geographical Information System) [28], providing a location and a series of optional additional information. The tool uses the values of the solar irradiance and of the external temperature (that influences the performance of the photovoltaic panels) to evaluate the energy generated during each hourly time-step. At the moment, the user is supposed to provide a properly-formatted input file containing the installation's production profiles. Anyway, a future development of the tool may foresee the possibility to communicate directly with PVGIS to automatically evaluate the profiles once that the location is provided by the user.

2.1.2 Storage system

There are many types of electric energy storage systems, each one with its own peculiarities making it more suitable for a given application. Generally, figures of merit upon which different storage technologies can be classified are: the energy and power densities, that influence the size of the system, the power rating and the time of charge/discharge. As to the latter, systems are usually divided into short, medium and long term storage, which are supposed to have charge/discharge phases, respectively, of few minutes, few hours and days or even months (seasonal storage systems). Of course, also the capital costs and the lifetime are to be taken into account when choosing the storage solution to employ. In Figure 2.4, some energy storage technologies are divided basing two figures of merit, the typical duration of the charge/discharge phase and the power rating, identifying possible applications for each one of them.

Electrochemical storage systems are among the most suited technologies for self-consumption in residential applications (see Figure 2.5), where usually charge/discharge times of few hours are needed [29]. Among these, lithium-ion batteries are the most employed in the domestic field, thanks to their relatively high power and energy densities, high round-trip efficiency (around 95%), small self-discharge and relatively long lifetime (in terms of number of cycles) [14]. Moreover, their modulability makes them suitable for a wide range of applications, from small electronics to electric mobility and large-size systems providing storage solutions for the transmission lines. For this reason, they have experienced large investments in Research&Development and are experiencing a continuous reduction in price [14, 29]. However, this technology has some drawbacks, such as the strong loss in performances and expected lifetime when overheating occurs [29].

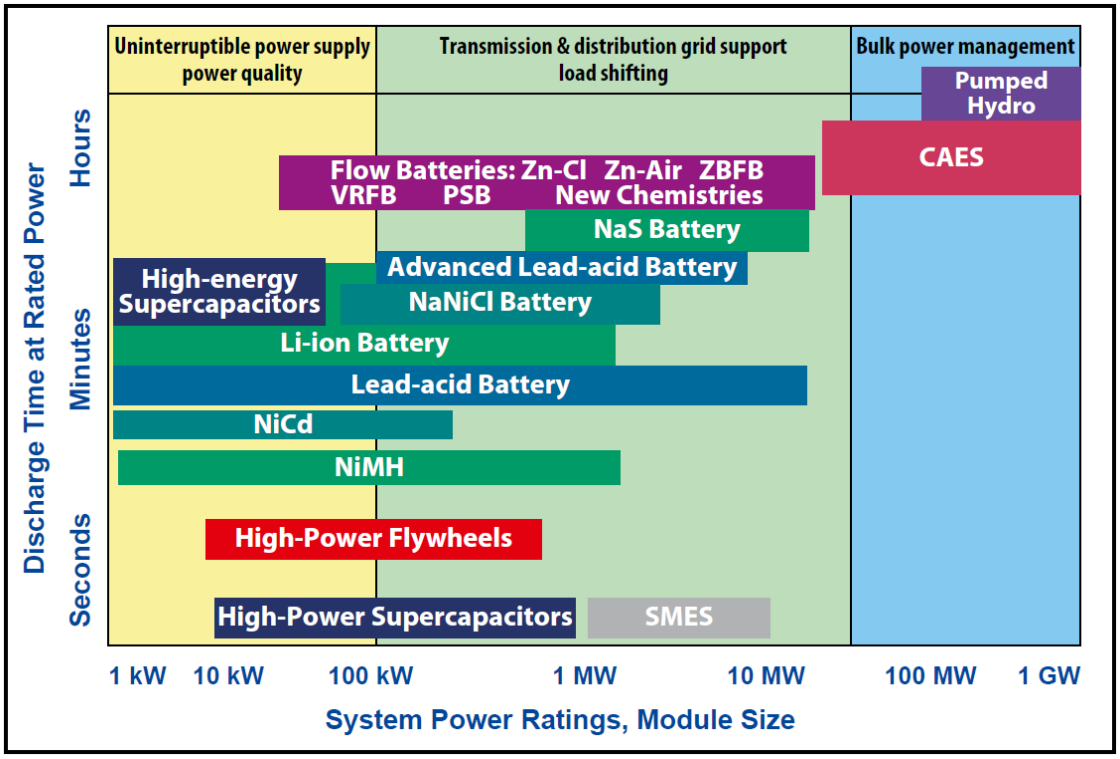


Figure 2.4: *Application fields of different storage technologies based on power ratings and discharge time. [Source: IRENA [29]]*

In the tool, a lithium-ion battery is considered as storage system. It is modelled as a "container" with a given capacity, where energy can be charged or discharged losing a part of it according to, respectively, the charge and discharge efficiencies (η_{bc}, η_{bd}). Moreover, a part of the energy stored in the battery is lost over time due to the self-discharge phenomenon (η_{sd}). The nominal capacity (CAP_{nom}), which coincides with the size in kWh, is not the total energy that is usually stored in the battery, due to the concept of the Depth of Discharge (DOD). In order to preserve the performance of the battery, indeed, full charge or discharge cycles are usually avoided, hence the full capacity battery of the battery is never deployed. In lithium-ion batteries, this useful capacity (CAP_{uf}) is generally not smaller than the 85 % of the nominal capacity [29]. The DOD is taken into account in terms of maximum

Figure 2.5: *Suitable applications for different storage technologies. [Source: IRENA [29]]*

and minimum States of Charge (SOC), that are defined as the ratio between, respectively, the maximum and minimum energy that is stored in the battery over its nominal capacity. Lastly, it should be noticed that the power that charges or discharges the battery is bounded to a maximum value ($P_{bc/d,max}$) . The latter can be evaluated, once that the minimum discharge time of the battery ($t_{bc/d,min}$) and the useful capacity are known, as follows:

$$P_{bc/d,max} = \frac{CAP_{uf}}{t_{bc/d,min}} = \frac{CAP_{nom} \cdot DOD}{t_{bc/d,min}} \qquad [\text{kW}] \quad (2.1)$$

2.1.3 Consumption units and electric grid

The consumption units, i.e. the households, are considered as 'passive' energy sinks, meaning that their power demand has to be fulfilled at any time. Also, the latter cannot be changed, for instance, by shifting in time the usage of some electric

appliances (loads). The consumption profile is considered for the whole aggregate, summing together the power demand from all the households.

The electric grid on the other hand, is simply modelled as a source/sink of energy, where the only limitation is on the maximum power that it can provide. Since the scheme is virtual, the grid fulfills the whole demand from the consumption units. The maximum power is the sum of the single contractual powers that can be delivered at the households' POD, which is usually equal to 3 kW in the residential context.

2.1.4 Reference year

All calculations are performed on a one-year basis, considering a number of typical days that are representative for the whole year. In general, two types of days are considered: a work day (week-day, in the following) and a weekend-day. This distinction is necessary because some appliances are used differently depending on the type of day, hence affecting the consumption profiles. Furthermore, it is assumed that the usage of some appliances changes according to the season, such as the air-conditioner, which is usually employed only in the summer months. Therefore, in the simulation of the consumption profiles, two typical days are considered for each season. As to the production from the renewable installation(s) instead, a different profile is used for each single month. Hence, when combining the production and consumption profiles for the evaluation of the configuration's performances, two typical days (week-day and weekend-day) are considered for each month. Conventionally, it is assumed that the first three months of the year (from January to March) are in the winter season, the next three (from April to June) in the spring season and so on. Moreover, when evaluating yearly energy values, the daily values in each typical day are to be multiplied by the total number of days of that type in

one year. This 'distribution' of days is defined considering a reference year, which starts on a Monday 1 January.

When performing the calculations, depending on the typical day that is being evaluated, different files/variables/positions in arrays are to be used. In order to let the code browse among the latter, each season/month/day-type is given an *'ID number'* and a *'nickname'*, which are, respectively, a unique number and a two/three characters string used to identify them. The elements defining the reference year are shown in Table 2.1, together with the distribution of days among the typical days. All these elements are to be used by the various methods that perform the calculations. For this reason, in the code, a `dictionary` (**auxiliary_dict**) is used to store them. The latter is defined in the `main` file and passed to the methods whenever they are called.

Table 2.1: *Elements used to define the reference year in the code. [Self-processing]*

Season			**Month**			**Days distribution**	
Name	**ID**[a]	**NN**[b]	**Name**	**ID**	**NN**	**wd**[c]	**we**[d]
Winter	0	w	**January**	0	jan	23	8
			February	1	feb	20	8
			March	2	mar	22	9
Spring	1	ap	**April**	3	apr	21	9
			May	4	may	23	8
			June	5	jun	21	9
Summer	2	s	**July**	6	jul	22	9
			August	7	aug	23	8
			September	8	sep	20	10
Autumn	3	ap	**October**	9	oct	23	8
			November	10	nov	22	8
			December	11	dec	21	10

[a] 'ID' is a unique number used to identify the season/month; [b] 'NN' stands for 'nickname', used to identify the season/month; [c] 'wd' is the nickname used for 'week-day'; [d] 'we' is the nickname used for 'weekend-day'.

2.2 Input data

As briefly mentioned, the tool does not require many inputs from the user. These can be divided into two categories: files that are supposed to be already in the *'Input/'* folder before running the code; parameters that the user can enter from keyboard once that the simulation has started. Everything else that is needed by the tool to perform the simulation is already provided, or simulated by the tool itself. In the following, all the input data are described in detail.

2.2.1 Input of parameters from the user

There is a number of parameters that are to be set in order to define a proper simulation setup. These can be divided into three groups:

- Simulation parameters;
- Setup of the simulation for the photovoltaic system;
- Setup of the simulation for the battery.

The first group includes all the parameters that define the characteristics of the configuration that is to be simulated (number of households, location, average energy class of the appliances, and so on), while the parameters in the second and third group are used to define the type of simulation to be performed. In more detail, these are used to create the ranges of possible sizes, respectively, of the photovoltaic installation and the battery system that are to be explored. These parameters are shown in Table 2.2, divided by group, together with a brief description and the default values that are applied. It should be noticed that the type of simulation

that is chosen affects the other parameters that can be defined about the size(s) to be evaluated, both for the photovoltaic and the battery.

Table 2.2: *Parameters that can be changed by the user using inputs from keyboard. [Self-processing]*

Group	Name	Value	Description
Simulation parameters	**n_hh**	2	number of households (units)
	location	north	location (north − centre − south)
	power_max	3	contractual power of each household (kW)
	en_class	A	energy class of the appliances ($A_{+++} \div D$)
	ftg_avg	100	average footage of the households (m^2)
	dt_aggr	60	time-step for the aggregation (min)
Simulation setup for the PV	**sim_type**	fixed	type of simulation ('fixed' size or 'parametric')
	size	2	(fixed) size of the installation (kW)
Simulation setup for the battery	**sim_type**	parametric	type of simulation ('fixed' size or 'parametric')
	size_min	1	minimum size (kWh)
	size_max	5	maximum size (kWh)
	n_sizes	5	number of sizes to be simulated

Every time that the code is run, the values of the parameters are read from CSV files stored in the folder *'Parameters/'* and printed to the user's screen. If any of these files are not found or cannot be loaded, the default values shown in Table 2.2 are applied. Anyway, the user can change the values of the parameters, after which the aforementioned CSV files are overwritten so that there is no need to updated them again in the future simulations. The user is provided an interactive way of updating the values of the parameters, using self-created methods that allow for inputs from keyboard. The latter are contained in the module `parameters_input.py`. Whenever the user wants to update the value of a given parameter, a whole expression (for example 'n_hh = 100') should be typed. This way, the user can update only the parameters of interest rather than running through all of them every time. If any

mistake is done in writing the parameter's name, the Levenshtein distance[1] is used to suggest the user the closest matching one. On the other hand, if any mistake is done in writing the parameter's new value, the correct data type and lower/higher limits or possible values are suggested.

After completing the updating phase, the new values are printed to the user's screen. The 'simulation parameters' are stored in a `dictionary` (**params**), which is shared among the various methods, while the parameters in the 'simulation setup' groups are used to create `lists`, containing the sizes of the photovoltaic system and of the battery that are to be simulated. Lastly, it should be noticed that, in order to lighten it, only the main parameters can be changed through this procedure. There are in fact other parameters that are more specific to certain methods and less likely to require to be changed by the user. Anyway, this can still be done but manually, operating on the code. These parameters are addressed later on, when describing the methods in which they are used.

2.2.2 Battery specifications

As previously mentioned, reasonable values for a typical lithium-ion battery are used by default in the tool, which are shown in Table 2.3. Nevertheless, the user can provide different values, which are found for instance in the data-sheet of a specific battery model. If this is the case, the user is supposed to overwrite the values stored in a CSV file named *battery_specs*, that is located in the *'Input/'* folder. Referring to the values shown in Table 2.3, it can be noticed that the battery can be charged to

[1]The Levenshtein distance is defined as the smallest number of elementary operations needed to change one string into another one. The elementary operations are: adding or removing a character and changing a character into another one. Every operation that is to be performed increases the distance by 1 [30].

its maximum capacity (full charge), but it cannot be fully discharged: the maximum Depth of Discharge is equal to 85%.

Table 2.3: *Typical values of the battery specifications for a lithium-ion technologies (used by default in the tool). [Self-processing]*

Name	Value	Description
SOCmin	0.15	minimum state of charge (-)
SOCmax	1	maximum state of charge (-)
t_bc_min	3	minimum time of charge/discharge (h)
eta_charge	0.98	charge efficiency (-)
eta_discharge	0.94	discharge efficiency (-)
eta_self_discharge	0.999	self discharge efficiency (-)

2.2.3 Production from the photovoltaic installation

The user is supposed to provide a proper CSV file where the installation's hourly production profiles are stored. According to the previous discussion, one profile for each month is needed. The user may choose to get the data from PVGIS, in which case the *hourly radiation data* tool can be used. In Figure 2.6 a screenshot of the online tool's interface is shown.

After having selected the geographical location, some other optional quantities can be specified, such as the mounting type, the slope and the azimuth. One can either choose to let the tool optimize the latter, if they are not constrained by the geometry of the roof where the installation is going to be located. Different databases of the solar radiation are present in the tool: the one used in the following is PVGIS-SARAH, which contains data from 2005 to 2016 for the main European areas. To get the hourly production from the installation, the *'PV power'* box should be ticked, choosing a peak power of 1 kW_p in order to get unitary values. The peak power is defined as the power that the photovoltaic array *«can produce under standard test*

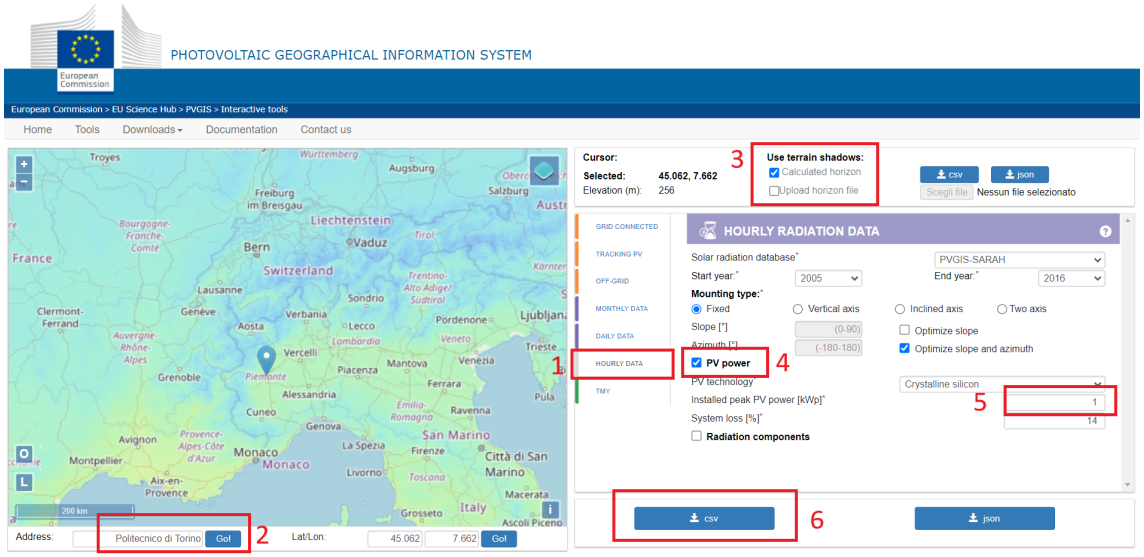


Figure 2.6: *Interface of the online tool PVGIS. [available at [28]]*

conditions, which are a constant 1000 W/m^2 *of solar irradiance in the plane of the array, at an array temperature of* 25 °C*»* [28]. When performing the simulation, the 'unit' values are multiplied by the actual size of the installation, in terms of peak power, to get the real production.

The hourly data, of all days of the selected years, can be downloaded as a CSV file. Anyway, the format of the latter is not very convenient for the subsequent calculation. Therefore, a module is provided (`pvgis_to_csv.py`) to automatically evaluate the hourly production profiles for a typical day in each month, averaging the values among the selected years. The module requires the CSV file downloaded from PVGIS to be located in the *'Input/'* folder and named *'PVGIS_Data'*. The properly-formatted CSV file created by the module is saved in the same folder as *'pv_production_unit'*.

In Figure 2.7 the hourly unit production profile from a photovoltaic installation in a location of latitude 45.062 and longitude 7.662 (*Politecnico di Torino*) is shown for each month of the year. The profiles show the typical bell-shape, with a higher

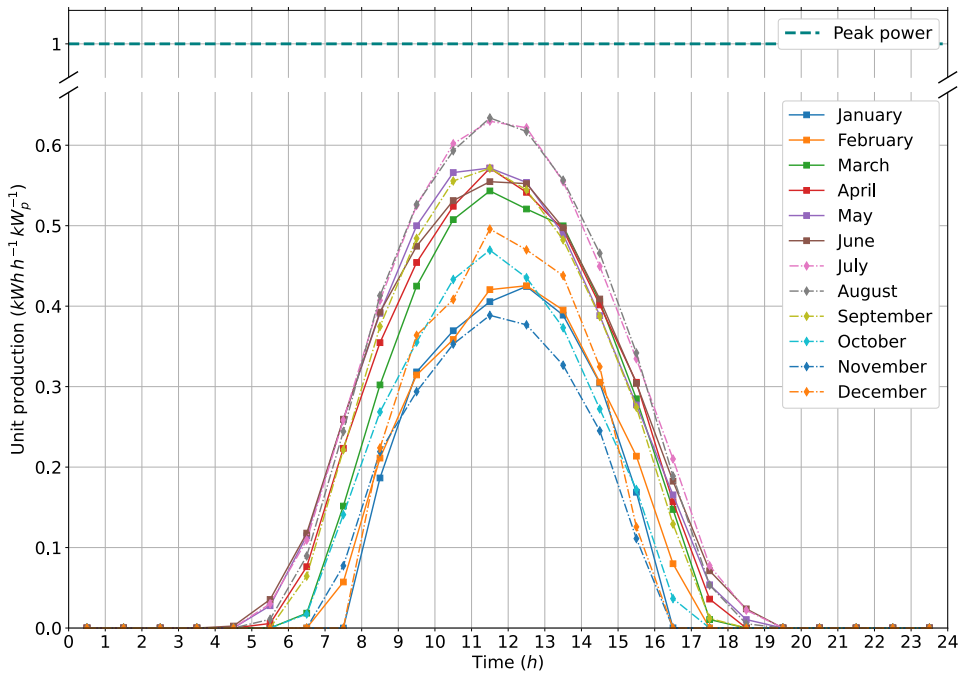


Figure 2.7: *Unit hourly production profiles in one typical day for each month (average from 2005 to 2016). [Self-processing of data from PVGIS [28]]*

production in the central hours of the day, when the radiation is larger. Similarly, the production increases moving from winter to summer months and then decreases again. Interestingly, the peak power is never reached (the maximum power is roughly 0.65 kW). This is the results of averaging, for each time-step in each month's profile, the values from a large number of days (all the days of the month, for several years), which surely have had different weather conditions. On the other hand, the peak production is reached when very specific conditions are realised (in terms of solar radiation and temperature). However, *instantaneous* peaks in the production would be toned down in case, due to the aggregation into hourly values.

2.2.4 Consumption of the aggregate of households

Knowing the yearly (or even monthly) energy consumption of a specific household is usually not an issue, since they appear in the energy bills. On the other hand, the consumption profiles during a number of typical days can be difficult to get, mainly due to the lack of proper metering instrumentation. In similar tools, the user is supposed to provide the consumption profiles in each typical day of calculation or, alternatively, the yearly/monthly values of the energy consumption. The latter are used to scale some normalised profiles (as in the optimisation tool from StoRES [31], for instance). These are 'typical' consumption profiles for a given type of customers, e.g. households, which returns a unitary energy consumption if integrated over one day (or one year, depending on which quantity is used to scale them).

The normalised profiles are, however, the result of an average among a large number of customers. Hence, it is unlikely that they can represent the high randomness and, usually, higher power peaks in the consumption profiles of a small number of customers. The peculiarity of this procedure is instead the simulation of the these profiles, for a given number of households, using probabilistic methods. Therefore, while the yearly energy consumption of the single households is, with a good approximation, constant, the instantaneous power demand changes every time that the profiles are simulated. As the number of households increases, however, the sum of all the consumption profiles tends to the typical shape for domestic customers, which usually has two power peaks, around noon and at evening.

Nonetheless, the user may want to simulate various configurations (with different sizes of the photovoltaic and/or the battery) using the same consumption profiles. Therefore, the latter are stored in CSV files (one file for each type of day of the week) in the *'Output/'* folder, any time that they are simulated. Once that the ag-

gregate's characteristics are defined (according to the procedure described in 2.2.1), the program starts searching in the folder to check if the profiles for the same aggregate have already been simulated. If so, the user is given the possibility to choose between using the available data or simulating new ones.

The procedure for the simulation of the consumption profiles of the aggregate of households is covered in detail in 2.3. Anyway, it should be recalled that the tool simulates the profiles in two typical days for each season. On the other hand, two typical days for each month are considered in the rest of the routine. Hence, to get the 'monthly' profiles, the profile in a given season is assigned to the first month of the latter, while interpolating between adjacent seasons for the other months, as shown in Figure 2.8.

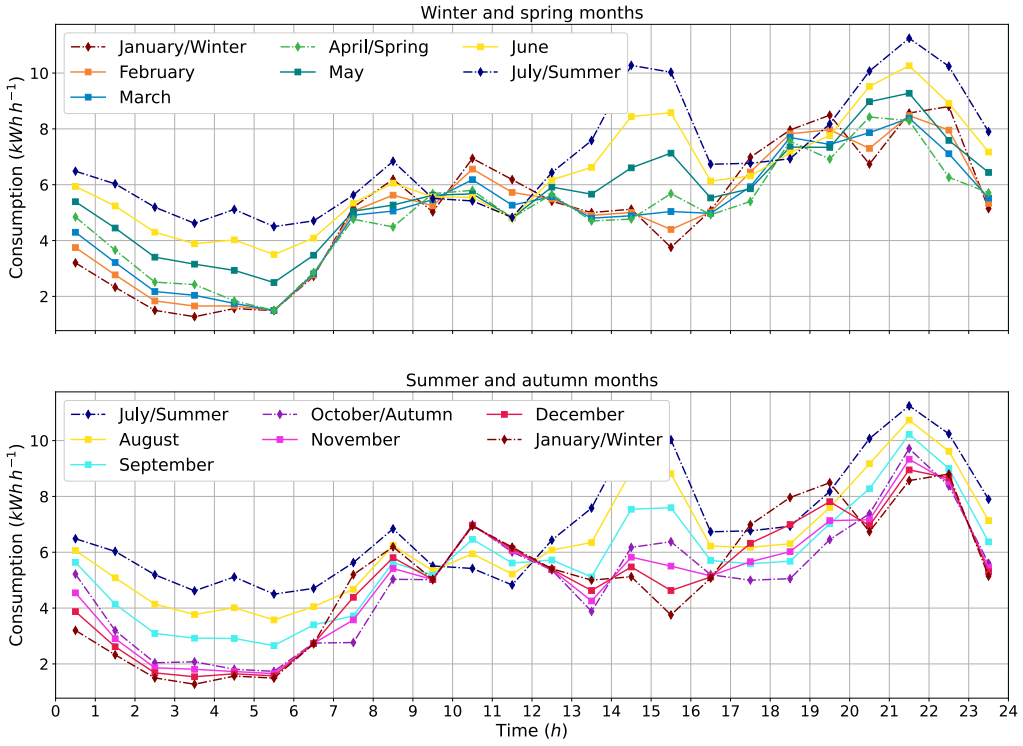


Figure 2.8: *Consumption profiles simulated for all seasons and interpolated into months. [Self-processing]*

2.3 Simulation of the load profiles of an aggregate of households

Before presenting the routine for the simulation of the *consumption profiles*, it should be noticed that the expression *load profiles* is mostly used in the following. The focus is indeed on the power demand from the electric appliances present in the households (loads). However, the two terms can be considered fully equivalent in the context of this work.

The routine follows a bottom-up approach in a step-wise fashion. In more detail, this means that the load profiles of the aggregate of households are obtained by simulating and adding together the load profiles of *each single household*. The latter are simulated considering the instantaneous power demand from *each electric appliance* that is found in the household. To do this, a probabilistic approach is used, generally drawing a time-instant in which the appliance is switched-on from a probability distribution for the usage of the appliance, derived from a series of actual measurements. The power demand of the appliance is added to the total load profile of the household, starting from the aforementioned time-instant, and for the whole *duration* of the appliance, namely the total time in which the latter is used during one day. Furthermore, energy classes[2] are used to change the yearly energy consumption of some appliances, which of course affects the appliances' load profiles.

Different load profiles are simulated for each season, considering two typical-days, a week-day and a weekend-day. The total time in each typical day, i.e. 1440 min,

[2]An average energy class is to be specified for the whole aggregate of households.

is discretised with a resolution of 1 min. Anyway, the results are aggregated with a different time-step, depending on the granularity needed for the further calculations and specified by the user (usually one hour). The total and average energy consumption from the appliances are also evaluated for the reference year.

2.3.1 Collection of data about the appliances

The routine relies on a set of data that were the result of a measurement campaign carried out by eERG (end-use Efficiency Research Group) in the early 2000s, named MICENE (Electric Energy Consumption Measurement). The objective of the study was to build a database of measurements of the power demand from a set of domestic appliances, thus providing a useful tool to be used in the context of energy efficiency and demand-side management in households. During the campaign 110 households from different regions of Italy were monitored for a period of at least three weeks in order to evaluate the energy consumption from the main electric appliances. The latter was measured with a time-step of ten minutes, resulting in a series of load profiles showing the average power demand during each 10 min interval. Of course, measurements from different days were used. Hence, the results for each category of appliances (e.g. dishwasher) were averaged, first for the single unit, among the different days of measurement, and afterwards among the all units monitored. This way, an *average daily load profile* was obtained for each category of appliances. Anyway, as a direct consequence of this operation, the resulting profile is a rather smooth curve, with relatively low peaks. The power demand from a single appliance, on the other hand, is usually edgy with high peaks. This difference stems from the fact the different units of the same appliance were switched-on at different moments depending on the day and on the household [32].

The appliances considered in the study were: dishwasher, washing machine, refrigeration and freezing equipment (fridge, freezer), electric boiler, audio-video and electronic devices (television, video-recorder, personal computer) and lighting. For some of them, different profiles were evaluated, depending on the type of day of the week. In Figure 2.9 the *average daily load profile* of the dishwasher is shown, for both a week-day and a weekend-day[3]. As to the lighting equipment, instead, different profiles were evaluated depending on the season (winter, summer or autumn/spring).

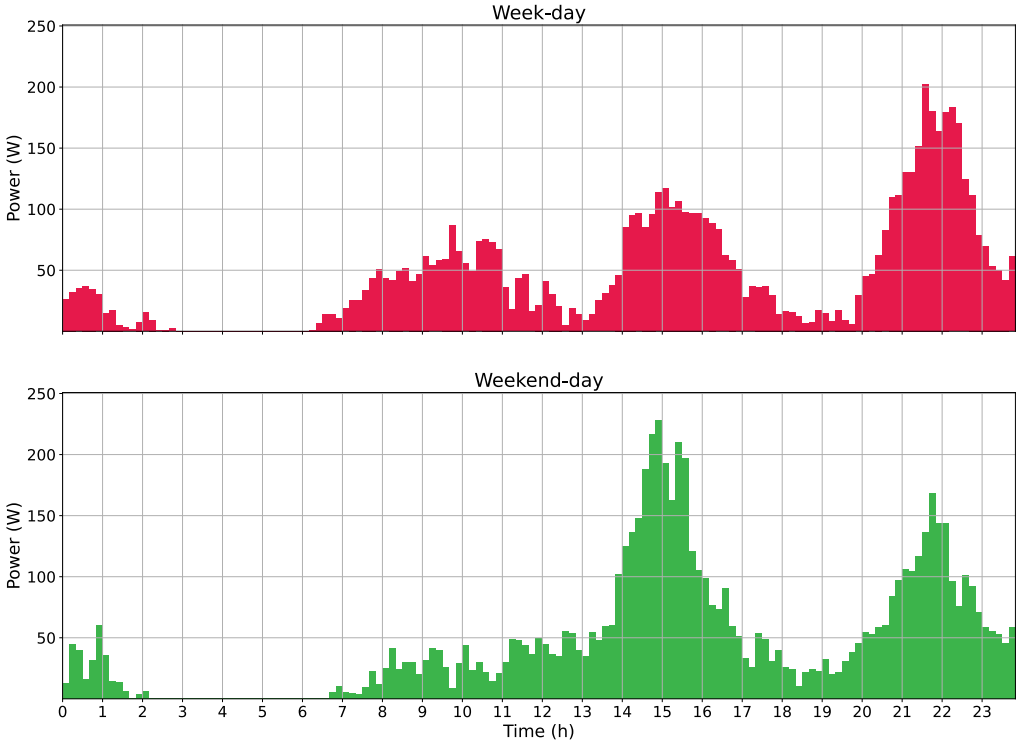


Figure 2.9: *Daily average load profiles for the dishwasher. [Self-processing of data from MICENE [32]]*

[3]As stated in [32], the curves are represented through bar graphs to *«highlight the temporal discretisation of the measured samples»*. The power that is here represented is indeed an average, uniform power over each 10 min interval.

In the report it is pointed out how the average energy consumption from the monitored households (3229 kWh) was relatively high, with respect to the Italian average (around 2400 kWh reported by [33] in 2009). This is due two different reasons. On the one hand, households with a large number of appliances were monitored in the study, to have a more representative sample. Hence, their total energy consumption was higher than the average. On the other hand, the appliances themselves had a quite high energy consumption. For example, the average yearly energy consumption of the monitored fridges was equal to 637 kWh/year, while for low-consumption fridges, it was roughly 200 kWh/year (stated by the producers) [32].

The *average daily load profiles* evaluated during the study are freely available at [34], as spreadsheets. Anyway, they are strictly valid for Italian households only, since the consumption habits can vary significantly between different countries. Anyway, the study was part of a wider project, EURECO, that covered four more European countries (Denmark, Greece, France, Portugal) for a total of 400 households monitored. The reader can find further information about the two studies at [34].

Other appliances that were not covered by [32] are also considered in the routine, such as the air-conditioner, electric and microwave ovens, tumble drier, vacuum cleaner, iron. For some of them, the same *average daily load profiles* as other ones have been assumed: as to the vacuum-cleaner and the tumble-drier, the one from the washing machine; as to the iron the one from the television; as to the hifi-stereo the one from the video-recorder and for the laptop the one from the personal computer. Of course, this introduces a certain approximation since these couplings are based on intuitive reasoning about the usage of the appliances rather than actual evidences. As to the air-conditioner and the electric and microwave ovens[4], an *average daily load*

profile has been built using the data from REMODECE (REsidential MOnitoring to Decrease Energy use and Carbon emissions in Europe) [35].

Some appliances (dishwasher, washing machine and tumble drier) do not have a constant power demand over their working cycles. The latter (duty cycle, in the following) is represented through typical diagrams, which have been found in [37] for the dishwasher and the washing machine. They are shown in Figure 2.10, where the instantaneous power demand during the working cycle over the maximum power is represented. As to the tumble drier, the same diagram as the washing machine has been assumed as a first approximation.

A number of additional data about the appliances (*attributes*, in the following) are needed: the time period in which it is averagely used during one day (namely, its *duration*); its yearly energy consumption and its distribution factor (or ownership rate) in different geographical locations. The latter is used to assess how each type of appliance is widespread among the population. It measures the number of units of a type of appliance that are found in a sample of households (over the total number of households) and can range between 0 and 1[5].

All these data were reported in [36] in tabular format. They were obtained performing a cross-reference of different sources, mainly ISTAT (the Italian National Institute of Statistics) [39], Enea [40] and the International Conference Eedal. An attempt has been made to trace back this sources in order to validate the data and, if possible, to update them to more recent values. This was not an easy task due

[4]The data used for the ovens are actually from [36], where it is stated they were taken from the REMODECE project's website, which however is no more available online. Different data have been found in the projects' report [35]. Anyway, the latter referred to a monitoring campaign performed in many European countries, therefore they have not been used.

[5]Actually, more units of the same appliance could be present in the same household, thus making the factor larger than 1. For example, [38] reported an average number of 1.8 televisions per household in Italy in 2003. Anyway, at the moment, the routine does not cover this case.

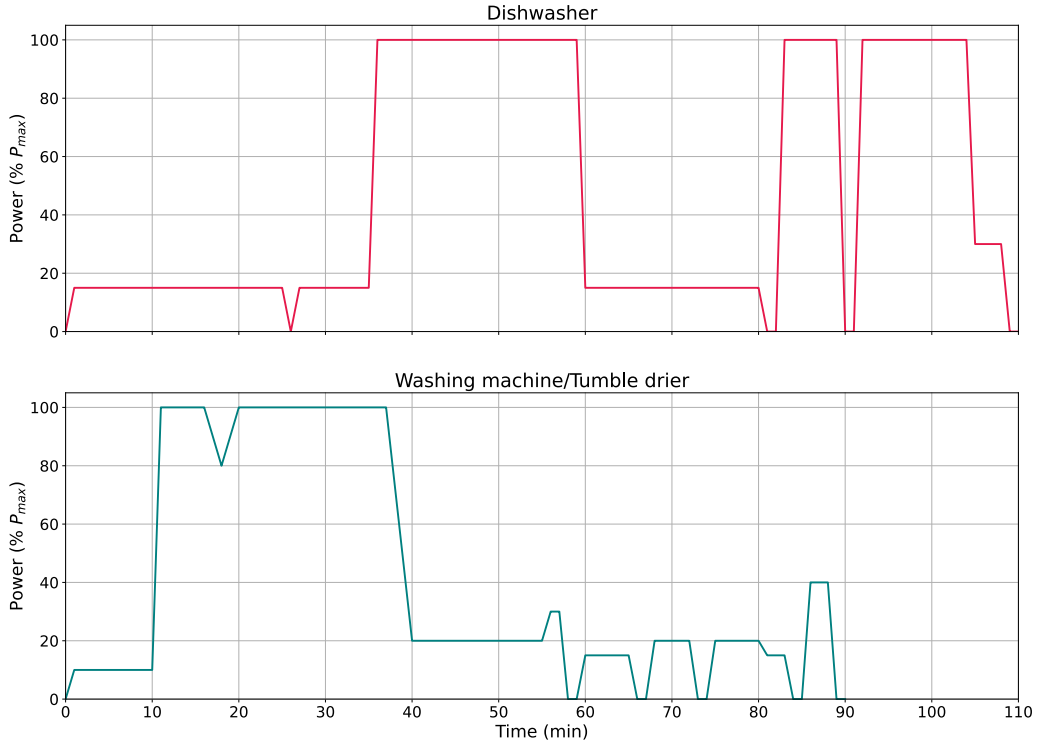


Figure 2.10: *Working cycle diagrams for some appliances. [Self-processing of data from CESI [37]]*

to the heterogeneity and the fragmentation of the sources, some of which are no more available online (e.g. the REMODECE project's website). Moreover, some data were from Italian studies, e.g. the ones from ISTAT or MICENE, while others were from studies involving many European countries, such as REMODECE. This might not be a relevant issue for those data that are more 'objective', such as the yearly energy consumption of the appliances according to the energy classes. Other factors, instead, are strongly influenced by the habits and customs of each particular population, e.g. the distribution factors and the average time in which an appliance is used during one day. In the light of these considerations, the following approach has been followed to validate all the data:

- Distribution factors: the priority has been given to data from Italian studies (such as ISTAT [39] or CESI [38]). When they were not available, the data reported in [36] have been employed directly.

- Average *duration*: in most of the cases, the data reported in [36] have been employed directly. Anyway, a comparison with data reported in [32], [40], [35] brought to similar results.

- Yearly energy consumption: when available, the data divided by energy class (from Enea [40]) have been employed. The missing data have been taken from MICENE [32], where the average yearly consumption of the monitored appliances was reported.

The distribution factors and average daily usage times are shown in Table 2.4, together with the average powers of the appliances. The latter provide just purely indicative values since the power demand usually changes significantly depending on the energy class. Some of these values were reported in various sources [35, 37, 38], while others have been evaluated starting from the available data about the yearly energy consumption and the appliances' average daily usage. On the other hand, for some appliances (mainly the electronic devices), the data about the daily usage time were difficult to find in any source. Hence the average powers reported in the sources and the yearly energy consumption have been used to evaluate them. The values of yearly energy consumption of the appliances (divided by energy classes, when available) are shown in Table 2.5.

Lastly, it should be noticed that the use of an appliance may vary depending on the season, according to the user's habits. For example, the air-conditioner is usually only used during summer (unless considering the switching to electric heating, which is not covered in the routine at the moment). Similarly, it is reasonable

Table 2.4: *Attributes of the appliances considered in the routine. [Self-processing of data from [35–39, 41]]*

Appliance	**Distr. North** (−)	**Distr. Centre** (−)	**Distr. South** (−)	**Avg power** (**W**)	**Time-on** (**min/day**)
Air-conditioner	0.32	0.25	0.32	1000	360[a]
Lighting	1	1	1	/	/
Fridge	1	1	1	150	1440
Freezer	0.28	0.26	0.21	200	1440
Electric oven	1	1	1	2000	60
Microwave oven	0.75	0.7	0.6	1100	11
Electric boiler	0.22	0.3	0.38	1255	200
Iron	1	1	1	1500	15
Vacuum cleaner	1	1	1	1300	12
Washing machine	1	1	1	1900	90
Dishwasher	0.5	0.5	0.3	2100	110
Tumble drier	0.1	0.1	0.1	1900	68[b]
Desktop computer	0.5	0.5	0.5	120	180
Television	1	1	1	200	240
Laptop	0.4	0.4	0.4	100	72
Hifi stereo	0.6	0.6	0.6	100	90
DVD reader/Videogame	0.8	0.8	0.8	80	180

[a] Evaluated considering that the appliance is only used in summer; [b] Evaluated considering that the appliance is not used in summer.

to assume that the tumble-drier is not used during summer, thanks to the higher external temperatures. For the same reason, it is likely that the fridge and freezer's consumption is larger in summer than in winter. Therefore, coefficients that modify the load profiles of the appliances according to the season (either increasing the power demand or the *duration* of the appliance, as discussed later on) are used. The latter are shown in 2.6.

In most cases, due to a lack of proper information, the coefficients have just been set to 1, meaning that the appliance's usage does not change with the season. In other cases, the coefficients have been given reasonable values. Anyway, these should be used carefully and, in the perspective of a future development of the routine, different ways of accounting for the different usage of the appliances according

Table 2.5: *Yearly energy consumption of the appliances under different energy classes (average values). [Self-processing of data from [36, 40]]*

Appliance	**Yearly energy consumption** $(kWh/year)$						
	$\mathbf{A_{+++}}$	$\mathbf{A_{++}}$	$\mathbf{A_{+}}$	**A**	**B**	**C**	**D**
Air-conditioner	320	550	700	850	920	980	1100
Lighting				427			
Fridge	130	170	240	309	406	530	640
Freezer	140	180	250	320	420	550	660
Electric oven	70	90	110	130	150	170	200
Microwave oven				33			
Electric boiler	450	550	900	1300	1500	1800	2600
Iron				135			
Vacuum cleaner	10	13	19	25	31	37	45
Washing machine	150	160	190	210	240	270	300
Dishwasher	220	240	270	300	350	390	420
Tumble drier	120	150	190	280	370	420	450
Desktop computer				132			
Television	65	85	119	170	227	255	283
Laptop				60			
Hifi stereo				55			
DVD player/Videogame				70			

Table 2.6: *Coefficients accounting for the user's seasonal behaviour. [Self-processing]*

Appliance	**Seasonal coefficients** $(-)$		
	Winter	**Spring/Autumn**	**Summer**
Air-conditioner	0	0	1
Lighting	1	1	1
Fridge	0.8	1	1.2
Freezer	0.8	1	1.2
Electric oven	1.1	1	0.9
Microwave oven	1	1	1
Electric boiler	1.3	1	0.7
Iron	1	1	1
Vacuum cleaner	1	1	1
Washing machine	1	1	1
Dishwasher	1	1	1
Tumble drier	1	1	0
Desktop computer	1	1	1
Television	1	1	1
Laptop	1	1	1
Hifi stereo	1	1	1
DVD player/Videogame	1	1	1

to the season might be foreseen. For instance, the switching from traditional to electric heating could be included in the routine, evaluating the power demand of

the air-conditioner through proper algorithms, which consider the variation of the Coefficient Of Performance of the appliance with the external temperatures.

2.3.1.1 Loading and processing of the data about the appliances

The data about the appliances described so far are stored in CSV files in the *'Input/'* folder, which are summarised in Figure 2.11. When simulating the load profiles, the files are loaded in the module `aggregate_load_profiler.py`, which serves as a sort of main module for this part of the simulation. The self-created methods contained in the module `datareader.py` are used to properly read the contents of the files.

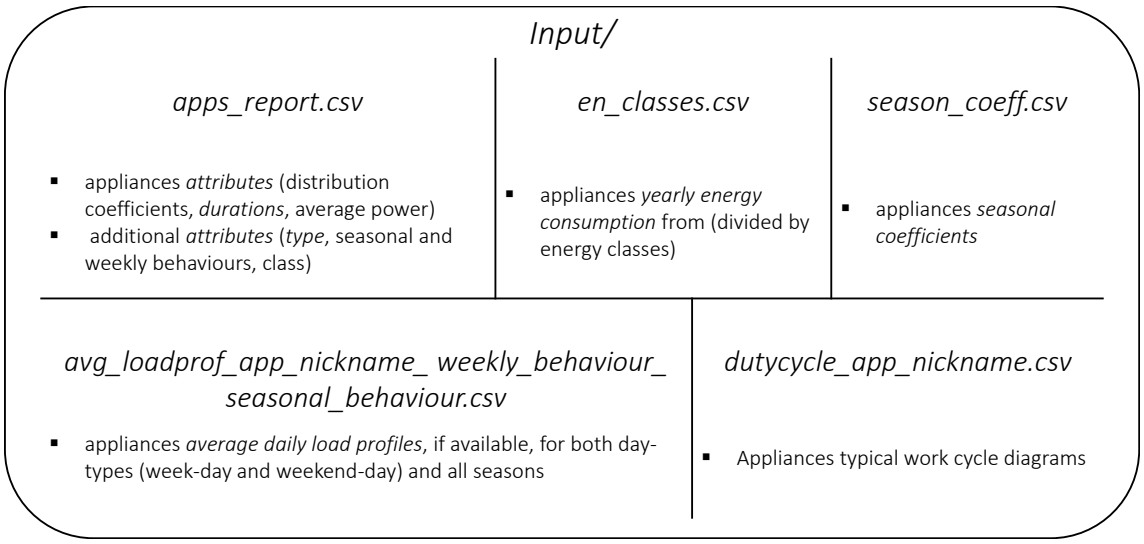


Figure 2.11: *Input files containing the data about the appliances. [Self-processing]*

The methods `read_appliances` and `read_enclasses` are used to load, respectively, the data about the appliances attributes and the data about the yearly energy consumption and seasonal coefficients. All these methods return a `2d-numpy.array` that reproduces the contents of Tables 2.4, 2.5, 2.6. Moreover, for each of them, a `dictionary`, which basically reproduces the first row and the first column of the tables, is used to properly slice the arrays and find the correct value of an *attribute* for

a given appliance, whenever it is needed. Other *attributes*, which are shown in Table 2.7, have been assigned to the appliances in order to enhance the automatisation of the routine. The ID number and nickname are used to identify each appliance in the various files and arrays of data (similarly as done for the seasons, months and days). The weekly and seasonal behaviour are used to identify if different *average load profiles* are to be used for the appliance, depending on the type of day (weekday/ weekend) and on the season. In more detail, the weekly/seasonal behaviour is set to, respectively, 'wde' or 'sawp' if only one profile is available for all types of days/seasons, and to, respectively, 'wd, we' or 's, w, ap' otherwise. The classes of the appliances instead are used to generate graphs about their energy consumption. The *types* of the appliances, instead, are discussed later on.

Table 2.7: *Appliances divided by type and class. [Self-processing]*

ID num.	**Name**	**Nick name**	**Type**	**Weekly behav.**	**Seasonal behav.**	**Class**
0	Air-conditioner	ac	continuous	wde	sawp	air conditioner
1	Lighting	lux	continuous	wde	s, w, ap	lighting
2	Fridge	frg	continuous	wde	sawp	refrigerating
3	Freezer	frz	continuous	wde	sawp	refrigerating
4	Electric oven	elo	uniform	wd, we	sawp	cooking
5	Microwave oven	mwo	uniform	wd, we	sawp	cooking
6	Electric boiler	bo	uniform	wde	sawp	boiler
7	Iron	ir	uniform	wde	sawp	cleaning
8	Vacuum cleaner	vc	uniform	wd, we	sawp	cleaning
9	Washing machine	wm	duty cycle	wd, we	sawp	washing
10	Dishwasher	dw	duty cycle	wd, we	sawp	washing
11	Tumble drier	td	duty cycle	wd, we	sawp	washing
12	Desktop computer	pc	uniform	wde	sawp	office
13	Television	tv	uniform	wde	sawp	entertainment
14	Laptop	lap	uniform	wde	sawp	office
15	Hifi stereo	hs	uniform	wde	sawp	entertainment
16	DVD reader/ Videogame	vr	uniform	wde	sawp	entertainment

The CSV files where the appliances' *average daily load profile files* are stored show the time, in hours, in the first column and the average power demand, in watts, in the second column. Their names are formatted as follows: *'avg_loadprof'* + *app_nickname* + *weekly_behaviour* + *seasonal_behaviour*. Similarly, the typical work cycle diagrams are stored in CSV files whose names are formatted as follows: *'dutycycle'* + *app_nickname*. They are loaded using the method `read_general`, which returns a `2d-numpy.array` (**data**), containing the time and power vectors, respectively on the first and second column. As to the *average daily load profiles*, the time-step of the data (generally, 10 min) is different from the one used in the routine (1 min), thus requiring an interpolation.

The interpolation of the profile can be seen as the transition from the *discrete*, where the power in each time-step is actually the ratio between the energy consumption in that time interval and the interval itself, to the *continuous*, where the power in each point is an instantaneous value. This also affects the methods that are used: for example when dealing with continuous, numerical integration is used instead of a simple summation over time. When it came to the interpolation methods, different options were available, such as the `numpy` built-in method `interp`, or `CubicSpline` and `interp1d`, from the `scipy.interpolate` package. As shown in Figure 2.12, all three methods have been tested.

On the one hand, `CubicSpline` provides a smoother profile, since it performs a polynomial interpolation, but also amplifies peaks and troughs. In some cases it even makes the power demand negative, which does not have a physical meaning and therefore requires to be saturated. Both `interp` and `interp1d`, instead, stick more to the original data. The latter, however, does not allow for a periodical interpolation over one day, which instead appears as the most reasonable option. As to the daily energy consumption (right side of the figure), it can be noticed that

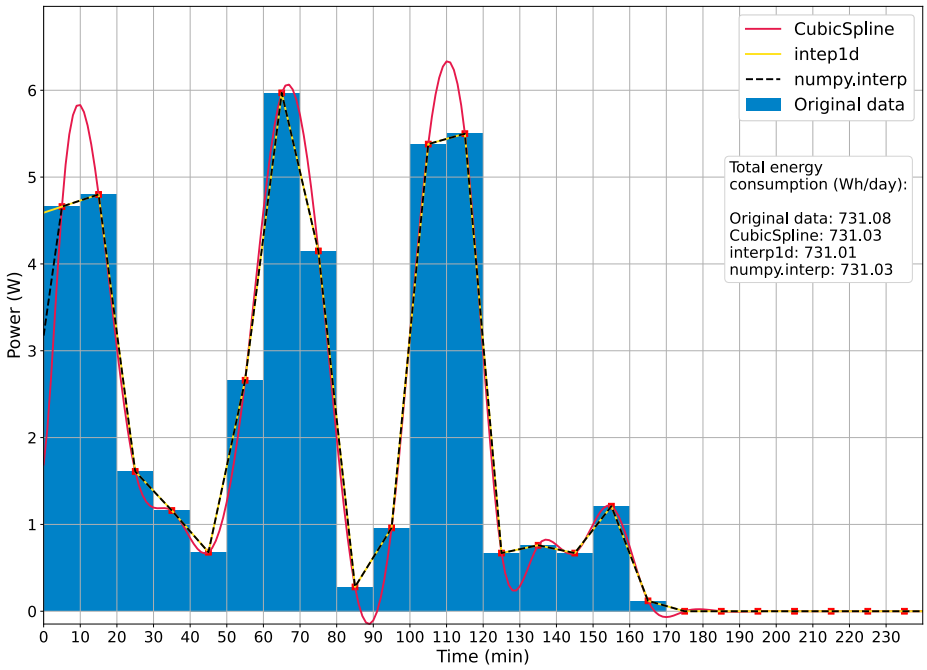


Figure 2.12: *Interpolated daily average load profiles for the washing machine. [Self-processing of data from MICENE [32]]*

in general it is not fully preserved after the interpolation, regardless of the method that is used. Anyway, the differences appear to be negligible. In light of the above considerations, `numpy.interp` has been chosen for the interpolation of the *average daily load profile*, using a period of one day, i.e. 1400 min.

After being loaded and, if needed, interpolated, the various arrays containing the *average daily load profiles* and the typical duty cycles are stored in two dictionaries. This is done to allow to automatically select the profile, depending on the appliance, the season and the day-type that are considered. They are stored, together with all the other variables containing the data about the appliances, in a new `dictionary` (**appliances_data**). The latter is passed to the various methods any time that these are called.

2.3.2 Methods for the simulation of the load profiles

The very core of this routine is the simulation of the instantaneous power demand, over one single day, from each appliance found in the households. After having obtained all the power demands, it is indeed quite straightforward to add them together into the household's and then the aggregate's load profiles. Of course, the simplicity of the last two operations is due to the fact that no loads-management strategy is applied and therefore the power demands from the appliances are taken as they are and *injected* into (i.e. added up to) the load profile of a household. Anyway, a slight control is operated at household level, in order to shift in time the usage of a given appliance, whenever the total power demand exceeds the maximum available power, namely the contractual power, as described later on. Considering that a bottom-up approach is followed in the routine, it seems appropriate to describe the simulation of the different-levels load profiles starting from a single appliance, then moving to the single household and finally to the aggregate of households.

2.3.2.1 Load profile of a single appliance

The method `load_profiler` (contained in a same-named module) is used for the simulation of a single appliance's load profiles during one single day. In the following, the term 'appliance' is used to address the specific one (e.g electric boiler), which is passed to the method in order to simulate its load profile. In the code, the appliance is identified by the variable **app**, which is a string corresponding to its name. The typical day that is to be simulated is identified instead by the season (**season**) and the day-type (**day**) that are passed to the method.

The following variables are introduced to discretise in time[6]:

- **time**, total time of the simulation, equal to 1440 min;
- **dt**, time-step, equal to 1 min;
- **time_sim**, a `1d-numpy.array` ranging from 0 to **time**, i.e. from 00:00 to 23:59, with a step of **dt**.

The method uses the variables contained in **appliances_data** to extract the appliance's attributes needed for the calculation, that are assigned to the following variables:

- **energy**, yearly energy consumption, in kWh/year, depending on the energy class if the data is available;
- **T_on**, *duration*, in min/day, i.e. the appliance's average time of usage during one day;
- **kk**, seasonal coefficient that takes into account the user's habits according to the season. In some cases, an appliance is not used at all during a season (e.g. tumble-drier during summer), therefore the coefficient is set to 0.

The appliance's load profile is stored in a `1d-numpy.array` (**load_profile**), which has the same size as the time-vector and is built in two different phases: shaping and scaling. After giving the proper shape to the instantaneous power demand from the appliance, indeed, the latter is to be scaled up or down so that, when integrating

[6]It is to be noticed that, similarly as the data about the appliances described in 2.3.1, these variables are actually declared and assigned in the module `aggregate_load_profiler.py` and shared as a `dictionary` (**time_dict**) among the modules.

it over one day and multiplying it by the number of days in one year, the result is equal to the appliance's yearly energy consumption. For a better understanding, the scaling of the load profile is described first, assuming that the instantaneous power demand has already been shaped.

First, some useful quantities are to be defined and evaluated. The *equivalent power* (**power**) of the appliance is here defined as the uniform value of the power demand that realises the same yearly energy consumption, if the appliance is used for its whole *duration* each day of the year. It can be evaluated as follows:

$$\mathbf{power} = \frac{(\mathbf{energy} \cdot 1000)/365)}{(\mathbf{T_{on}}/60)} \qquad [\mathrm{W}] \quad (2.2)$$

In general, this value does not represent the instantaneous power demand itself, since it might not be constant over the appliance's *duration*. It is used instead to adjust the power demand to the yearly energy consumption. To do this, a normalised instantaneous power, $P_{norm}(t)$ is first evaluated. Calling the instantaneous power demand $P(t)$ (shaped but not scaled yet), then $P_{norm}(t)$ can be evaluated as follows:

$$P_{norm}(t) = P(t) \cdot \frac{\int_0^{\mathbf{time}} dt}{\int_0^{\mathbf{time}} P(t)\, dt} \qquad [\text{-}] \quad (2.3)$$

The multiplication between the normalised instantaneous power and the *equivalent power* results in an adjusted instantaneous power demand, $P_{adj}(t)$, that realises the correct yearly energy consumption over one year. It is evaluated as follows:

$$P_{adj}(t) = P_{norm}(t) \cdot \mathbf{power} \qquad [\mathrm{W}] \quad (2.4)$$

$$\text{so that : } \left(\int_0^{\mathbf{time}} P_{adj}(t)\, dt \right) \cdot 365/1000 = \mathbf{energy}$$

In the following, whenever the operation of «adjusting the power demand to the yearly energy consumption» is mentioned, it refers to the sequence of Equations (2.2), (2.3), (2.4). In the routine, this is implemented in a very easy way, using `numpy`'s ability to perform operations between arrays. All the integration operations are performed using the method `numpy.trapz`.

All the elements needed for scaling the appliance's load profile have now been introduced. Still, the power demand from the appliance has not been shaped yet. To do this, the appliance's *average daily load profile* (derived as discussed in 2.3.1), is used. Similarly as the other data of the appliance, the latter is selected from **appliances_data**, depending on the season and the day-type and assigned to a `1d-numpy.array` (**avg_load_profile**). It is then used differently depending on the *type* of the appliance. To perform the calculation in a convenient way, indeed, the instantaneous power demand from the appliance is approximated. Three possible shapes that it may assume have been identified and the appliances have been divided in as many types (all the appliances' *types* are shown in Table 2.7):

- *Continuous*: the appliance is used constantly throughout the day (fridge, freezer), or the appliance can be repeatedly switched-on/off without having a well-defined pattern in the power demand (air-conditioner, lighting). Keeping track of this behaviour in a simple way would be a tough task, therefore it is assumed that the appliance is used throughout the whole day, with a given shape for the instantaneous power demand.

- *Duty cycle*: the power demand is not constant throughout the appliances' *duration* but it follows some well-defined patterns, modelled through the typical work cycle diagrams (washing machine, dishwasher, tumble drier).

- *Uniform*: the power demand can be assumed to be constant all over the *duration* of the appliance.

Different routines are followed to shape the appliance's instantaneous power demand, according to its *type*.

Appliance of *continuous* type

The *average daily load profile* is directly used to shape the instantaneous power demand throughout the day. Therefore the array **load_profile** is set equal to **avg_load_profile** and it is adjusted to the yearly energy consumption and multiplied by the seasonal coefficient **kk**. The lighting represents an exception, since its yearly energy consumption does not depend on the energy class (due to a lack of data). Anyway, its *average daily load profile* was evaluated for a number of houses whose square footage was around 100 m^2, therefore the power demand is corrected by a factor that takes into account the actual average square footage of the houses (**k_ftg** = **ftg_avg**/100). Afterwards, **load_profile** is returned.

Appliance not of *continuous* type

First, the power demand from the appliance during its *duration* is built. The latter is improperly[7] called 'duty cycle' in the following. It is identified by one array for the time, ranging from 0 to **T_on** with a step of **dt**, and one for the power, which has the same size as the former. On the contrary, the load profile is the power demand from the appliance during the *whole day* of simulation. The latter is therefore set

[7]It is an improper designation since for an appliance of *uniform* type, the power demand is constant over its *duration*.

equal to the duty cycle when the appliance is on and it is equal to 0 during the rest of the day:

$$P_{lp}(t) = \begin{cases} P_{dc}(t), & \text{when the appliance is on} \\ 0, & \text{when the appliance is off} \end{cases}$$

Where P_{lp} and P_{dc} are, respectively, the power demands in the load profile and in the duty cycle.

Appliance of *duty cycle* type The shape of the duty cycle is given by the typical work cycle diagrams. Anyway, the values of the power are adjusted to the yearly energy consumption.

Appliance of *uniform* type A uniform duty cycle is built starting from the *equivalent power* (**power**) and the *duration* ($\mathbf{T_{on}}$). Anyway, in order to account for variability in the user's behaviour, a new *duration* is drawn from a normal distribution centred in $\mathbf{T_{on}}$, with a given standard deviation (**devsta** set to 2 min). The displacement of the new *duration* from the original one is constrained by a percentage tolerance (**tol** equal to 15 %). This is implemented using the method `numpy.random.normal`, that is re-called until the drawn *duration* falls under the tolerance range. Lastly, the *duration* is multiplied by the factor **kk**, thus decreasing or increasing the usage of the appliance, according to the season.

Whether the appliance is of the *duty cycle* or *uniform* type, it is not used throughout the whole day, therefore a time instant in which the appliance is switched-on is to be evaluated. This is done by drawing a time instant from a probability distribution for the usage of the appliance. The latter is shaped using the *average daily load*

profile. First, the cumulative probability is evaluated by integrating the probability distribution over time and normalising the result, as follows:

$$p_{cum}(t) = \frac{\int_0^t p_{dens}(t)\,dt}{\int_0^{\mathbf{time}} p_{dens}(t)\,dt} \tag{2.5}$$

Where p_{cum} and p_{dens} are, respectively, the cumulative probability and the probability density of usage of the appliance during one day. The normalisation is needed because the *average daily load profile* is not actually a probability density but rather an instantaneous power (measured in watts). This operation is performed using the `scipy.integrate` built-in method `cumtrapz`.

Afterwards, a random probability ranging between 0 and 1 is extracted from a uniform distribution, using the method `numpy.random.rand`. The element in the vector **time_sim** corresponding to the element in the cumulative probability vector that equals the random probability can be considered as the instant when it is most probable that the appliance is on, i.e. when it is exactly at half its *duration*. The switch-on instant is therefore evaluated subtracting half the appliance's *duration* to this time-instant. The probabilistic procedure used to draw the random instant is shown in Figure 2.13, where if a very large number of random probabilities were to be extracted, the distribution of the instants would have had the same shape as the *average daily load profile*.

Starting from the switch-on instant, the duty cycle is *injected* into the appliance's load profile, that is then returned. This operation is implemented using the `numpy` built-in method `roll`, which is able to shift an array by a proper number of indices: the duty cycle is therefore injected at the beginning of **load_profile** and rolled until it starts in correspondence of the switch-on instant.

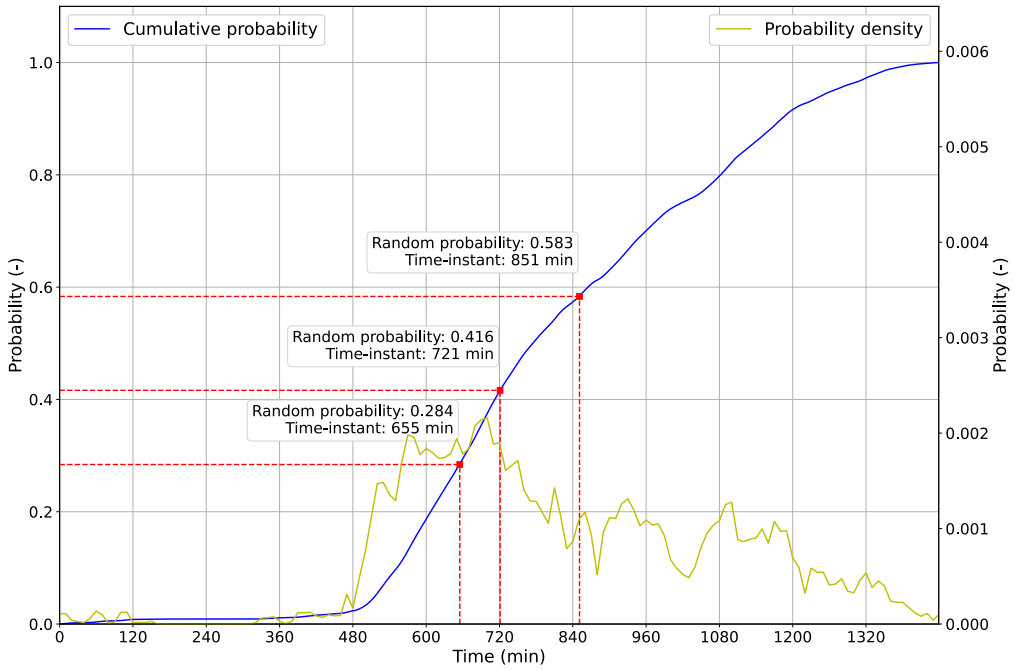


Figure 2.13: *Cumulative probability of usage of an appliance. [Self-processing of data from MICENE [32]]*

In Figure 2.14 the load profiles simulated for one appliance of each *type* are shown. Interestingly, for appliances not of *continuous* type, the switch-on instant may happen to be at the end of the day. In this case, the duty cycle (uniform or not) is partly injected in the beginning of the day. It should also be noticed that, if the daily energy consumption of each appliance, shown at the top-right corner of Figure 2.14, is multiplied by the number of days in one year[8], the resulting yearly energy consumption is consistent with the values shown in Table 2.4.

[8] As long as air-conditioners are involved, one should remember that they are used, conventionally, 100 days/year.

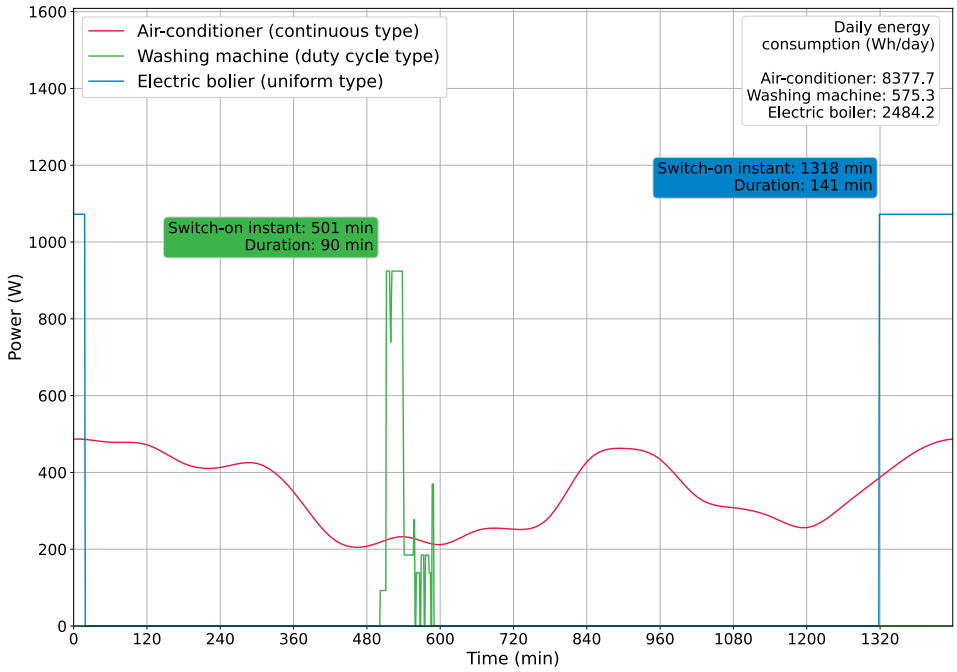


Figure 2.14: *Load profiles simulated for appliances of the three different types (summer, week-day, energy class A). [Self-processing]*

In Figure 2.15 the a large number of load profiles has been simulated for three different appliances (one for each *type*). The average load profile obtained from the simulation is compared to the *average daily load profile*, showing the ability of tool to re-create it. Of course, while the shape of the former only depends on the distribution of the switch-on instants and on the *duration* of the appliance, the scale is affected by the value of the yearly energy consumption, therefore a proper energy class is to be chosen in order to get similar values of the power demand. Lastly, the algorithm used in the method `load_profiler` is summarised in Figure 2.16.

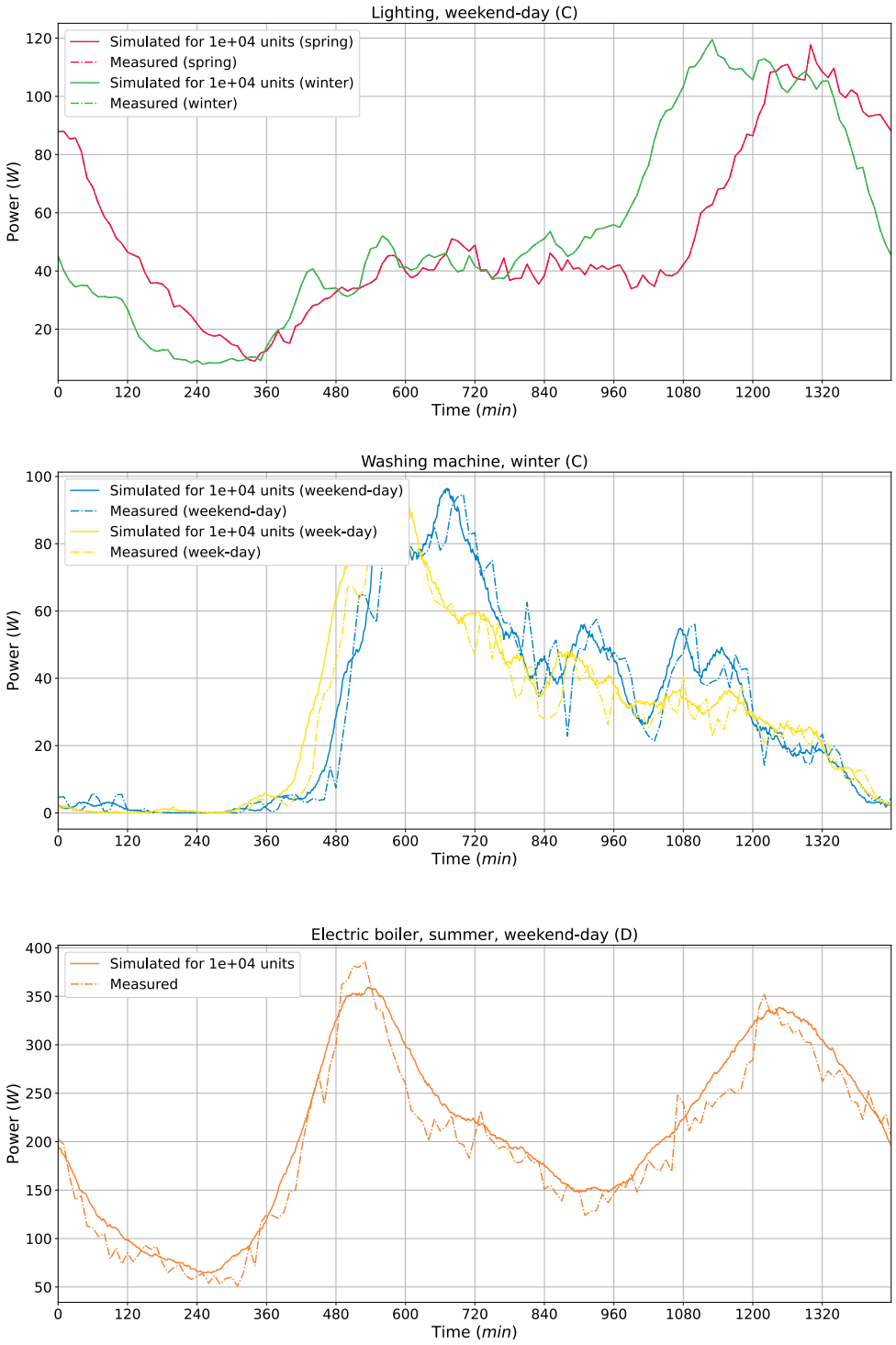


Figure 2.15: *Average load profile of different appliances resulting from a large number of simulations, compared with the average daily load profiles. [Self-processing]*

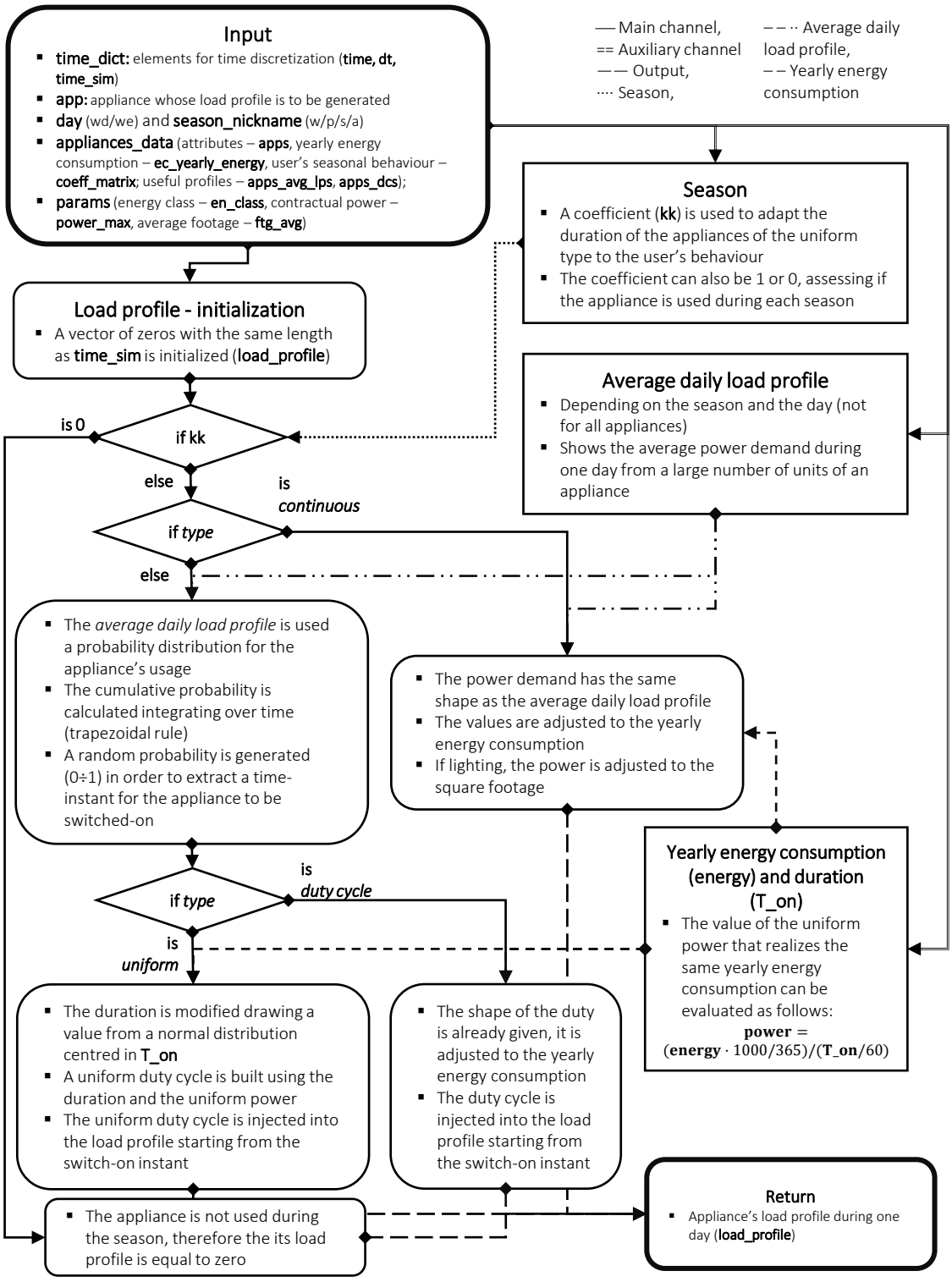


Figure 2.16: *Algorithm used to evaluate the appliance's load profile in the method* `load_profiler`. *[Self-processing]*

2.3.2.2 Load profile of a single household

The method `house_load_profiler`, from the same-named module, is used to simulate the load profile of a single household in a given typical day. Similarly as before, in the following the term 'household' is used to address the specific household on which the calculation is being performed. In addition to the data and parameters needed for the execution of `load_profiler`, the method needs the **apps_availability** vector as input. The latter is used to specify which appliances are found in the household, among the ones considered in the routine.

To evaluate the household's load profile, a vector of zeros of the same size as the time-vector (**house_load_profile**) is initialised and the the one-day power demands from the appliances are added to it, one at a time, as they are simulated by `load_profiler`. Of course, this is done only for the appliances that can be found in the household, i.e. those whose corresponding position in **apps_availability** is equal to 1. At this stage, a slight control is operated, if adding the current appliance's load profile causes the total power demand from the household to exceed the contractual power (**power_max**) at least in one instant. To do this the usage of the appliance is postponed by a proper time-step[9] until the total power demand is always smaller than the maximum power or until a maximum number of tries is reached. This is easily implemented using the method `roll` inside a 'while' loop.

Before moving to the next one, the daily energy consumption from the current appliance is evaluated integrating its load profile over the day. This value is stored in a `1d-np.array` (**energy**), whose size is equal to the number of appliances. It should be noticed that the appliances are run through in a particular order, to reduce

[9] At the moment, this is evaluated by averaging the total time in which the power demand exceeds the maximum power and the *duration* of the appliance.

the possibility that multiple postponements are required (which would distort the real usage of the appliances and increase the computational time). Therefore, the *continuous* type ones are evaluated first, since they cannot be postponed. The other appliances are evaluated starting from the highest-power-demanding ones. After simulating all the appliances' load profiles, if the power demand of the household still exceeds the maximum power, it is just saturated to this value. After this operation, the method returns both the **house_load_profile** and **energy** arrays.

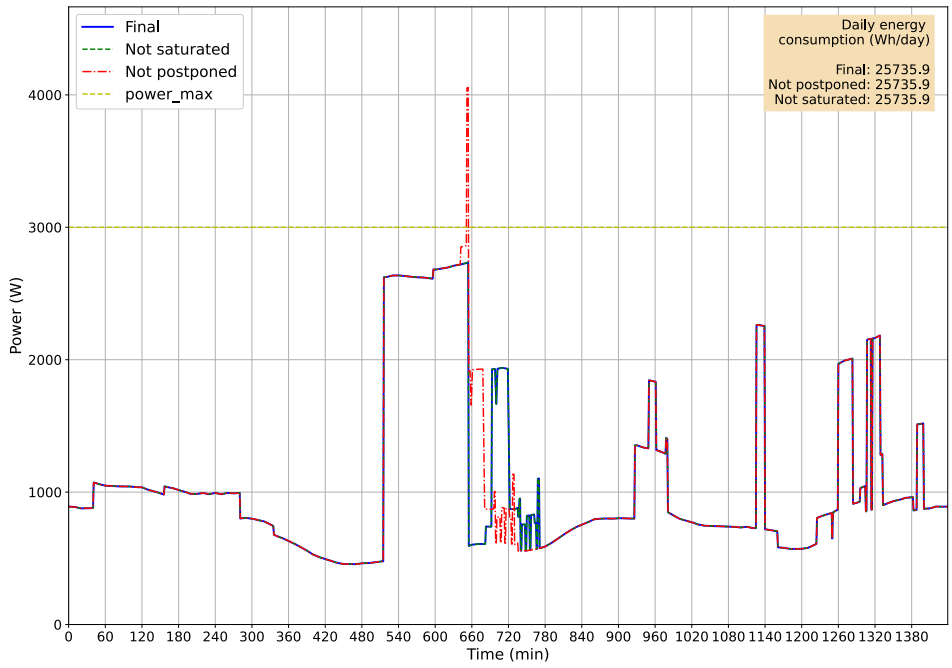


Figure 2.17: *Load profile simulated for a household, showing the postponing of an appliance due to having exceeded the maximum power. [Self-processing]*

The load profile simulated for a household where all the appliances are found is shown in Figure 2.17. The highest-consumption energy class (D) has been chosen on purpose, to increase the power demand from the appliances and have more chances

that the total power demand exceeded the maximum limit. Indeed, the usage of at least one appliance has been postponed (for better viewing only the last one to be postponed is shown). It can be noticed that, thanks to this operation, no saturation of the load profile has been required. As to the total energy consumption, as one could expect, it is not affected by the postponing the usage of the appliances (top-right corner of the figure). The algorithm implemented in the method `house_load_profiler` is summarised in Figure 2.18.

2.3.2.3 Aggregation of the load profiles

As previously mentioned, the method `aggregate_load_profiler` is used as a sort of main for this part of the simulation. In the latter, the load profiles of all the households in the aggregate are simulated for all the typical days. It should be recalled, indeed, that the previous methods simulate the load profile, either of a household or an appliance, during just one day.

The first step is to build the **apps_availability** matrix, whose columns represent the vectors showing the availability of the appliances in the households. The matrix is built running through the categories of appliances and assigning, for each category, the total number of units found in the aggregate to a randomly selected sample of households. The former is evaluated multiplying the distribution factors, shown in Table 2.4, by the number of households. The method `random.random_sample` is used instead to generate the random sample of households, which is returned as a `list`, containing the households in the sample. The latter is used to properly slice and fill the **apps_availability** matrix.

The households' load profiles can now be evaluated, one typical day at a time, using the method `house_load_profiler`. The profiles of the households are firstly stored separately, as columns of a `2d-np.array`, for the subsequent processing of

the results. As previously mentioned, for the calculations of the configuration's performances, a coarser time-resolution than the one used for the simulation of the profiles is fine. Therefore a new time-vector is created (**time_aggr**), which has a step of **dt_aggr**. Hence, after having simulated the profiles of the households in a given typical day, they are given the new time-step, using the self-built method `aggregator` (from the module `profile_aggregator_trapz.py`). The latter receives the original profile(s) and time-discretisation and the new time-step as inputs. These elements are used to create the new profile(s), which have the size as **time_aggr**. For each profile, a uniform value of the power (P_{aggr}) is evaluated, for each time-step in **time_aggr**, so that it realises the same energy consumption as the original profile (P) in the corresponding time-period:

$$P_{aggr}(t) = \frac{\int_t^{t+\mathbf{dt_{aggr}}} P(t)\,dt}{\mathbf{dt_{aggr}}},\ \forall t = 0 : \mathbf{dt_{aggr}} : \mathbf{time} \tag{2.6}$$

Afterwards, the profiles of all the households are summed together into the aggregate's total load profile (**aggregate_load_profile**). The total load profiles for all the typical days are stored in a `3d-np.array` (where the axes represent, respectively, the seasons, the one-day time-discretisation and the day-types), which is returned. The algorithm used in the method `aggregate_load_profiler` is summarised in Figure 2.19.

Further processing of the results

In addition to the total load profiles, other quantities are evaluated to provide more insight on the results, such as the average and the instantaneous medium, maximum and minimum load profiles. The last two are evaluated as quantile (respectively, the 90^{th} and 10^{th} household) in order to disregard very high or very low values of the

power. These profiles of interest are stored in properly-sized arrays which can be used to create a number of CSV files (one for each season and day-type) and figures (one for each season) where the profiles in the week-day and weekend-day are shown in separate subplots.

The daily energy consumption returned by `house_load_profiler` (for each household, divided by appliances) can be used together with the reference year discussed in 2.1.4 to evaluate seasonal and yearly values. To do this, the energy consumption from the appliances during each typical-day is multiplied by the number of days of that type that are present in the year (according to the days distribution in the reference year). The total energy consumption, divided by appliances and households, can be saved in separate CSV files for the different seasons. These values can also plotted in bar graphs (both the yearly and seasonal total consumption, divided by appliances), as well as the average yearly energy consumption from each category of appliance. The latter is evaluated taking into account the total number of units of each appliance that are found in the aggregate, rather than the number of households. Furthermore, the appliances are divided into classes, and the energy consumption from each class over the total energy consumption is represented in a pie plot. All the files and figures generated are stored in the *'Output/'* folder, respectively, in the *'Files'* and *'Figures'* sub-folders, where another sub-folder is created *ad hoc* for the simulation. It should be noticed that the user is asked whether to create and save these files and figures or not, since the process can be quite time-consuming[10]. As shown later on, this is done using some variables (flags), which are set either to 1 or 0, depending on the choice of the user.

[10]It usually takes 5-10 seconds, while the simulation of the aggregate's load profiles, for a number of households between 10 and 100, can take 0.1 - 2 seconds.

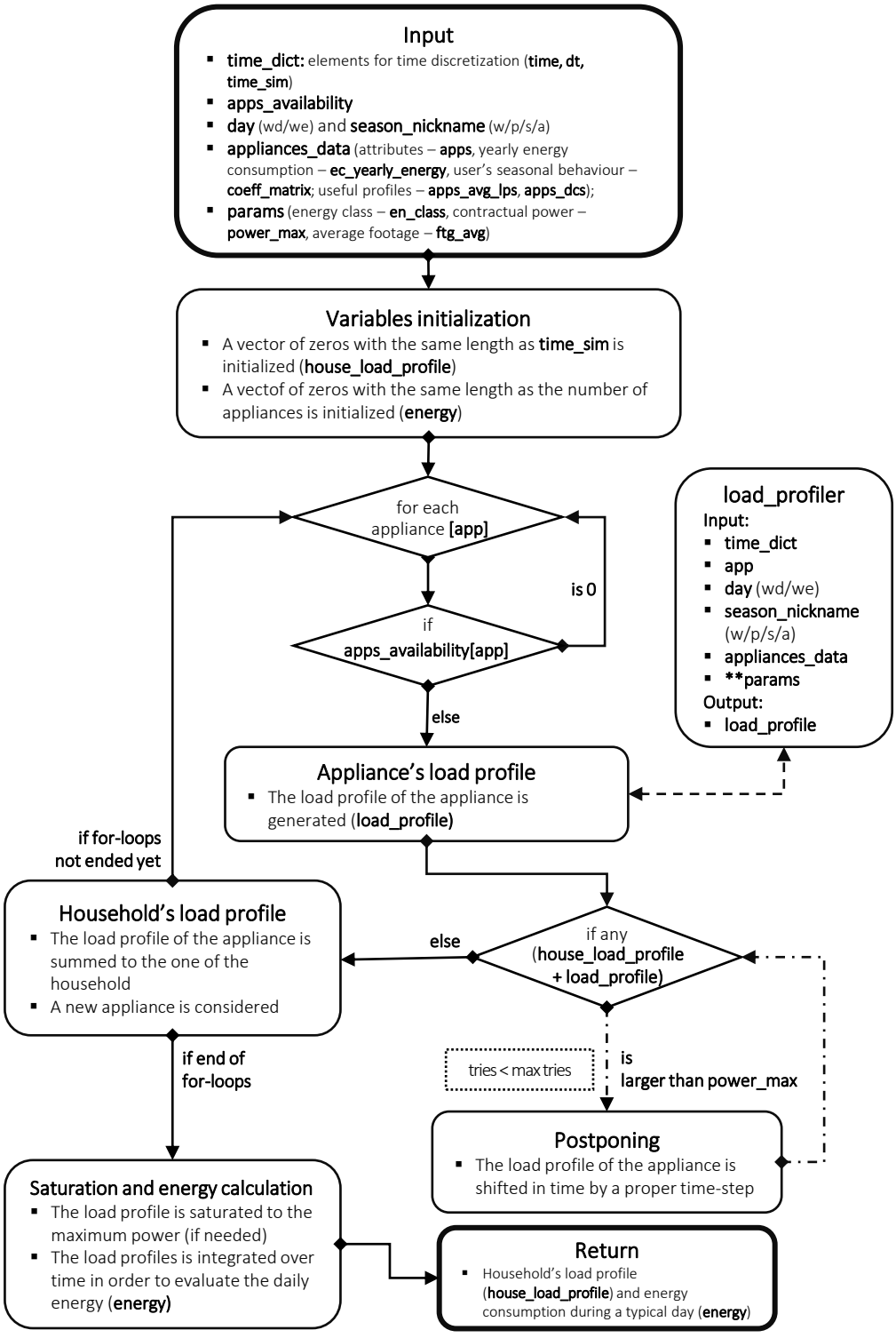


Figure 2.18: *Algorithm used to evaluate a household's load profile in* `house_load_profiler`*. [Self-processing]*

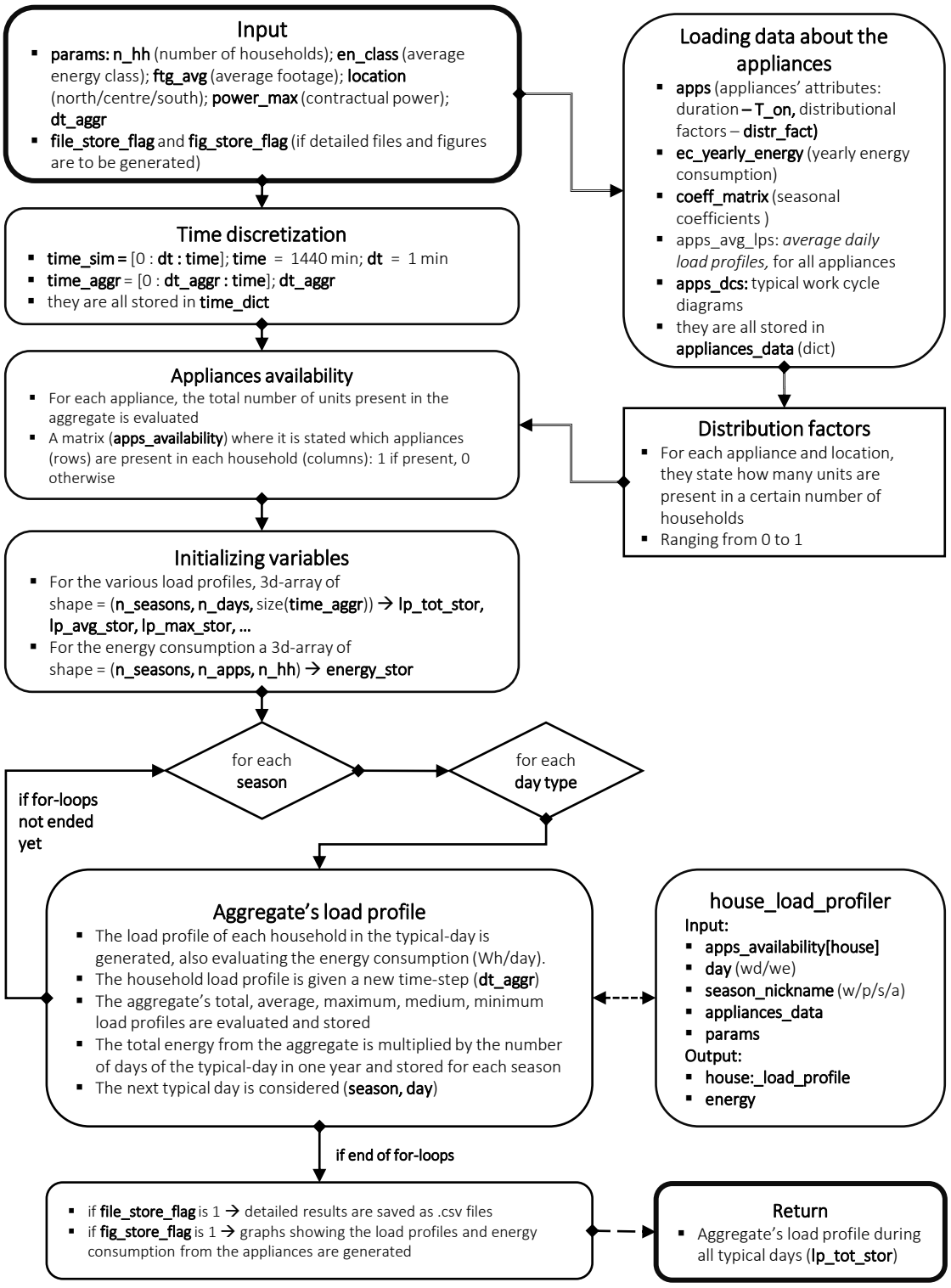


Figure 2.19: *Algorithm used to evaluate the aggregate's load profiles in* `aggregate_load_profiler`*. [Self-processing]*

2.4 Optimisation and evaluation

Having obtained the aggregate's consumption profiles during all typical days of simulation (see 2.2.4 for the conversion of seasonal data into monthly data), they can be used, together with the unit production profiles from the photovoltaic installation, to evaluate the performances of the configuration. It should be recalled that, for a given aggregate (identified by the number of households, the location, the energy class, etc), different sizes of both the photovoltaic system and the battery can be analysed, depending on the types of simulation (**sim_type**) that have been chosen by the user (see 2.2.1).

Each combination between the size of the photovoltaic system and of the battery identifies one configuration that is to be simulated and evaluated. Depending on the number of configurations to be evaluated, different results are provided. When a parametric analysis is chosen (whether on the photovoltaic, the battery or both) indeed, a preliminary evaluation is performed, and the different configurations are compared on a yearly basis. Whenever only one configuration is to be evaluated (fixed-size analysis on both the photovoltaic system and the battery) instead, more detailed results are provided. The latter include monthly values about the performances and the power fluxing optimised during each typical day. Hence, a flag is used (**fixed_analysis_flag**) to let the methods know which results to return. This is equal to 1 when there is only one configuration to analyse, 0 otherwise.

It should be recalled that, from this moment on, the calculations are performed with a different time discretisation than the simulation of the load profiles. The time-step that is used now is indeed the one provided by the user during the phase of input of parameters (**dt_aggr**), which is usually equal to one hour, in compliance with the provisions adopted in the Decree-Law 162/2019. It should also be noticed

that, while previously the reference time-units were minutes, from now on, hours are considered for sake of simplicity. The vector of time used in this part of the routine (**time_sim**) is hence created, using the new time-step. Moreover, all the elements used to discretise in time are converted into hours and stored again in **time_dict**. The latter is shared among the various methods that are going to be used. While the consumption profiles have already been given the new time-step (see 2.3.2.3), the unit production profiles, which rely on hourly data, may need to be interpolated, in the case **dt_aggr** is not 1 h.

2.4.1 Evaluation of the shared energy

As previously mentioned, the performances of each configuration (identified by the aggregate, the size of the photovoltaic and the one of the battery) are evaluated for one year. This is done by the method `shared_energy_evaluator`, from a same-named module, that receives the consumption and the unit production as inputs, as well as the information about the sizes, which are stored in a `dictionary`, **technologies_dict**. The unit production and the consumption profiles during all typical days are stored in a `dictionary` as well (**input_powers_dict**). As to the former, the unit values, in kW/kW_p, are to be multiplied by the size of the photovoltaic installation, in kW_p, in order to get the actual production. The method evaluates and returns a series of energy values during one year, divided by months (grid feed and purchase, battery charge and discharge, shared energy). First, these quantities are evaluated for each typical day in terms of powers (uniform values during each time-step), which are then multiplied by the time-step and summed together to get daily energy values and multiplied by the number of days in one year of each typical day. Furthermore, if the analysis is of *fixed-size* type for both the

photovoltaic system and the battery (**fixed_analysis_flag** is equal to 1), detailed results about the power flows during all typical days are returned as well.

In order to fully understand the power flows that are realised in the configuration, it is useful to distinguish between what actually happens and what can be considered conceptually happening. It has been mentioned several times that all the production is fed into the grid, while all the consumption is taken from the grid. The same holds for the energy that flows, respectively, out of and in the battery. This is summarised in Figure 2.20, in which the electric line where all these flows meet is collapsed into a fictitious electric node.

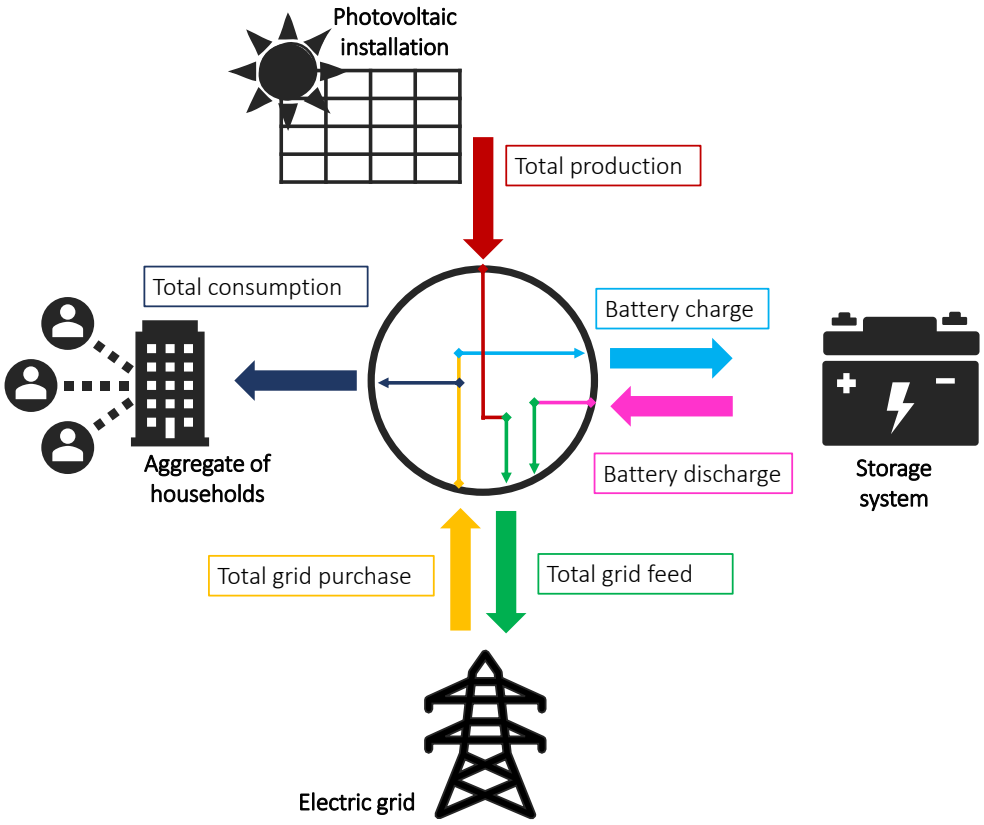


Figure 2.20: *Actual power flows realised in a scheme for virtual sharing of electricity. [Adapted from RSE [25]]*

Anyway, from a conceptual (and regulatory) point of view, the production from the photovoltaic installation is virtually self-consumed. Therefore, it can be imagined that it is used to fulfill the demand from the users participating in the configuration. When there is an excess, it can be used to charge the battery (**battery_charge**) or fed into the grid (**grid_feed**). Similarly, when the production is smaller than the consumption, the latter can be fulfilled discharging the battery (**battery_discharge**) or purchasing power from the grid (**grid_purchase**). The strength of such scheme, legitimised by the proximity of the consumption units, is that the virtual sharing of energy can be technically compared to a physical self-consumption. Hence, the 'conceptual' power flows represented in Figure 2.21 should be regarded as a reference point for the following.

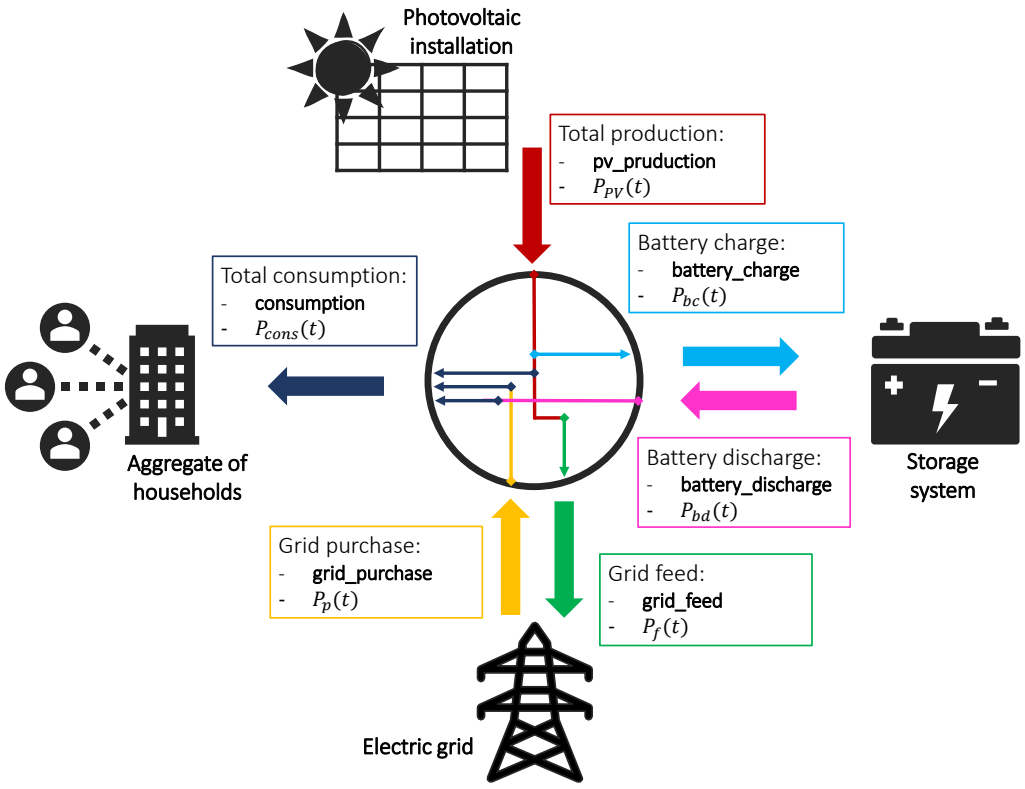


Figure 2.21: *Conceptual power flows realised in a scheme for virtual sharing of electricity. [Adapted from RSE [25]]*

The battery's usage strategy is defined in each typical day of simulation by optimising these power flows in order to minimise the exchanges with the grid (feed and purchase) or, alternatively, to maximise the shared energy. According to the Decree-Law 162/2019, the latter is evaluated as the minimum in each hour between the injections and the withdrawal into/from the grid, even through the storage system. Therefore, the following definition applies:

$$E_{sh}(t) = \min\left((P_{PV}(t) + P_{bd}(t) - P_{bc}(t),\ P_{cons}(t)) \cdot dt\right) \tag{2.7}$$

Where: E_{sh} is the shared energy (or power, if the time-step is not considered), P_{cons} and P_{PV} are, respectively, the total consumption and production during each time-step, P_{bd} and P_{bc} the battery discharge and charge powers.

On the one hand, it is quite straightforward to identify the withdrawals from the grid as the aggregate's consumption. On the other hand, the definition of the injections requires some more elaboration: the production and the discharge of the battery are effectively injected in the grid; the subtraction of the battery charge instead can be explained by the need to account for the energy stored in the battery just once. The latter aspect is better clarified later on, when the results provided by the tool are discussed.

2.4.2 Optimisation at the electric node

The objective here is to present the equations that characterise the optimisation of the power flow, mainly focusing on the physical and mathematical meaning of these equations, in the context of optimisation problems. Afterwards, the implementation in the code is briefly described. Optimisation problems usually involve a set of decision variables, which can be changed within a certain domain in order to reach

an objective. The domain is defined, in the first instance, by the values that the variables can assume. The former is further narrowed down by the constraints added to the problem, which can either be equations or inequalities, thus identifying the so-called *'feasible region'*. The objective can either be to minimise or maximise the objective function, which is a combination of the decision variables. The general formulation of an optimisation problem is conventionally written as follows:

$$\begin{cases} \min f(\boldsymbol{x}) \\ \boldsymbol{x} \in \Omega \\ g_i(\boldsymbol{x}) = 0,\ i = 1, \ldots, N \\ h_j(\boldsymbol{x}) \leq 0,\ j = 1, \ldots, M \end{cases} \tag{2.8}$$

Where: $\boldsymbol{x} \in \mathbb{R}^n$ is the set of decision variables, Ω the domain to which the variables belong, $f : \mathbb{R}^n \to \mathbb{R}$ is the optimisation function, $g_i : \mathbb{R}^n \to \mathbb{R}$ and $h_j : \mathbb{R}^n \to \mathbb{R}$ are, respectively, the N equality and M inequality constraints.

A particular type of optimisation problems is represented by *Linear Programming*, where both the constraints and the objective are linear functions [19]. The general mathematical formulation can be written as follows:

$$\begin{cases} \min \boldsymbol{c}^T \cdot \boldsymbol{x} \\ \overline{\boldsymbol{A}} \cdot \boldsymbol{x} = \boldsymbol{b} \\ \boldsymbol{x} \geq 0 \end{cases} \tag{2.9}$$

Where: $\boldsymbol{c} \in \mathbb{R}^n$, is used to define the objective, that is in this case a *linear combination* of the optimisation variables ($\boldsymbol{x} \in \mathbb{R}^n$), $\overline{\boldsymbol{A}} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$ are used to define the set of equality constraints and $\boldsymbol{x} \geq 0$ is used to define the domain of the optimisation variables (which in this case are positive definite) [42].

The problem in exam falls into a particular branch of *Linear Programming*, namely the Mixed-Integer-Linear-Programming (MILP), where the decision variables can either be integer or real numbers [19]. The power flows are indeed real variables, which are considered positive definite. Furthermore logic variables, which can either be 1/'True' or 0/'False', define, at each time-step, the *state* of the various *'actors'* of the optimisation, (i.e. the feed of excess energy into the grid, or the discharge of the battery). The approach of logic variables is used to represent the mutual-exclusivity of some of these *actors* [19], as discussed later on.

The equations that are going to be described involve a number of variables and parameters that are reported, respectively, in Table 2.8 and Table 2.9. Anyway, the names and symbols used for the power flows are the same as shown in Figure 2.21, while the parameters about the battery have already been introduced in Table 2.3, where the default values are shown (see 2.2.2).

Table 2.8: *Variables used for the optimisation of the power flows. [Self-processing]*

Name[a]	**Symbol**[b]	**Description**
pv_production	P_{PV}	Production from the photovoltaic installation(s)
consumption	P_{cons}	Consumption from the aggregate of households
	Optimisation variables	
grid_feed	P_f	(Excess) power fed into the grid
grid_purchase	P_p	(Deficit) power purchased from the grid
grid_feed_state	s_f	State of the grid feed (1|0)
grid_purchase_state	s_g	State of the grid purchase (1|0)
battery_charge	P_{bc}	Power that charges the battery
battery_discharge	P_{bd}	Power that is discharged from the battery
battery_energy	E_{stor}	Energy that is currently being stored in the battery
battery_charge_state	s_{bc}	State of the battery charge (1|0)
battery_discharge_state	s_{bd}	State of the battery discharge (1|0)
shared_power	P_{sh}	Power shared from within configuration

[a] 'name' refers to the `Python` variable; [b] 'symbol' to the notation used in the following equations.

Table 2.9: *Parameters used for the optimisation of the power flows. [Self-processing]*

Name[a]	Symbol[b]	Description
pv_size	$P_{PV,max}$	Peak power of the photovoltaic installation (kW)
battery_size	CAP_{nom}	Size of the battery or nominal capacity (kWh)
battery_charge_max	$P_{bc,max}$	Maximum power that can charge the battery (kW)
battery_discharge_max	$P_{bd,max}$	Maximum power that can discharge the battery (kW)
grid_feed_max	$P_{f,max}$	Maximum power into the grid (kW)
grid_purchase_max	$P_{p,max}$	Maximum power from the grid (kW)
Battery specifications (see also Table 2.3)		
SOCmin	SOC_{min}	Minimum state of charge
SOCmax	SOC_{max}	Maximum state of charge
t_bc_min	$t_{bc/d,min}$	Minimum time of charge/discharge
eta_charge	η_{bc}	Charge efficiency (-)
eta_discharge	η_{bd}	Discharge efficiency (-)
eta_self_discharge	η_{sd}	Self discharge efficiency (-)

[a] 'name' refers to the `Python` variable; [b] 'symbol' to the notation used in the following equations.

The electric node represented in Figure 2.21 sees a number of incoming and outgoing powers flows. The conservation of energy at the node requires the equivalence between everything that flows in and out of the node, at each time-step:

$$[P_{cons}(t) + P_{bc}(t) + P_p(t) - P_{PV}(t) - P_{bd}(t) - P_f(t)] \cdot dt = 0, \quad \forall\, t = 0 : \mathbf{dt\_aggr} : \mathbf{time} \tag{2.10}$$

Similarly, the energy conservation is to be imposed on the battery. Anyway, in this case, energy can also be stored:

$$E_{stor}(t + dt) = E_{stor}(t) \cdot \eta_{sd} + \left(P_{bc}(t) \cdot \eta_{bc} - \frac{P_{bd}(t)}{\eta_{bd}} \right) \cdot dt, \quad \forall\, t = 0 : \mathbf{dt\_aggr} : \mathbf{time} - \mathbf{dt\_aggr} \tag{2.11a}$$

$$E_{stor}(0) = E_{stor}(t) \cdot \eta_{sd} + \left(P_{bc}(t) \cdot \eta_{bc} - \frac{P_{bd}(t)}{\eta_{bd}} \right) \cdot dt, \quad t = \mathbf{time} \tag{2.11b}$$

A different equation is imposed in the last time-step, Equation (2.11b), where the quantity of energy stored in the battery at the end of the day is constrained to be equal to the energy stored at the beginning of the day.

Furthermore, the energy that can be stored in the battery is bounded not only by the size of the battery but also by the Depth of Discharge (see 2.1.2 and 2.2.2):

$$E_{stor}(t) \leq E_{stor,max}, \forall\, t = 0 : \mathbf{dt\_aggr} : \mathbf{time} \tag{2.12a}$$

$$E_{stor}(t) \geq E_{stor,min}, \forall\, t = 0 : \mathbf{dt\_aggr} : \mathbf{time} \tag{2.12b}$$

Where: $E_{stor,max}$ and $E_{stor,min}$ are evaluated multiplying the size of the battery by, respectively, the maximum and minimum states of charge.

The power deployed by the battery, either during the charge or discharge phase is also constrained by the upper-limit evaluated through the Equation (2.1), which is reported here for the sake of readability:

$$P_{bc/d,max} = \frac{CAP_{uf}}{t_{bc/d,min}} = \frac{CAP_{nom} \cdot DOD}{t_{bc/d,min}} \tag{2.1 revisited}$$

$$\begin{aligned} P_{bc}(t) &\leq s_{bc}(t) \cdot P_{bc/d,max}, \\ \forall\, t = 0 &: \mathbf{dt\_aggr} : \mathbf{time} \end{aligned} \tag{2.13}$$

$$\begin{aligned} P_{bd}(t) &\leq s_{bd}(t) \cdot P_{bc/d,max}, \\ \forall\, t = 0 &: \mathbf{dt\_aggr} : \mathbf{time} \end{aligned} \tag{2.14}$$

In the two equations (2.13), (2.14) the logic variables describing the *state* of the charge and discharge of the battery are introduced. It can be noticed how they are used to activate or deactivate the related variable: with respect to the charge of the battery, for example, if the state (s_{bc}) is zero, the power (P_{bc}) can only be zero.

The *states* are useful to impose some physical constraint, such as the fact that the battery cannot be charged and discharged at the same time. This is done by imposing the sum of the two *states* to be either 0 (they are both deactivated) or 1 (only one of the two is active):

$$s_{bc}(t) + s_{bd}(t) \leq 1, \forall\, t = 0 : \mathbf{dt\_aggr} : \mathbf{time} \tag{2.15a}$$

$$s_{bc}(t) + s_{bd}(t) \geq 0, \forall\, t = 0 : \mathbf{dt\_aggr} : \mathbf{time} \tag{2.15b}$$

Similarly, the grid feed and purchase are imposed to be mutually-exclusive. Theoretically, this is not strictly true, since different PODs are used for the consumption units and the production installation. Anyway, the constraint is imposed in order to avoid ineffective situations where, for instance, the consumption is fulfilled purchasing energy from the grid even if there is some production from the installation, which is instead fed into the grid. These situations are usually avoided once that the optimisation objective is set but the following constraints are used for extra safety:

$$s_p(t) + s_f(t) \leq 1,\ \forall\, t = 0 : \mathbf{dt\_aggr} : \mathbf{time} \tag{2.16a}$$

$$s_p(t) + s_f(t) \geq 0,\ \forall\, t = 0 : \mathbf{dt\_aggr} : \mathbf{time} \tag{2.16b}$$

The grid feed/purchase powers are also bounded by maximum values, which depend on the contractual powers set on the PODs. It might be argued that these constraints are avoidable in this case. The consumption profiles are in fact already saturated to the maximum power (see 2.3.2.2), while it is reasonable to suppose that the photovoltaic installation's POD is sized to accommodate the maximum production. Nonetheless, the two constraints are necessary for imposing the mutual-

exclusivity between the grid feed and purchase:

$$P_f(t) \leq s_f(t) \cdot P_{f,max}, \quad \forall\, t = 0 : \mathbf{dt\_aggr} : \mathbf{time} \tag{2.17}$$

$$P_p(t) \leq s_p(t) \cdot P_{p,max}, \quad \forall\, t = 0 : \mathbf{dt\_aggr} : \mathbf{time} \tag{2.18}$$

In addition, it is necessary to avoid that the battery is charged purchasing energy from the grid or that it is discharged to feed energy into the grid. Therefore, both the battery charge and grid feed powers are bounded by the power that is produced by the photovoltaic at each time-step:

$$P_{bc}(t) \leq P_{PV}(t), \quad \forall\, t = 0 : \mathbf{dt\_aggr} : \mathbf{time} \tag{2.19}$$

$$P_f(t) \leq P_{PV}(t), \quad \forall\, t = 0 : \mathbf{dt\_aggr} : \mathbf{time} \tag{2.20}$$

Lastly, the evaluation of the shared power, whose definition is provided in Equation (2.7), requires a more elaborate discussion. The function 'minimum' is in fact not linear and needs to be linearised. One approach that can be followed in such cases requires the use of two auxiliary quantities: a logical variable (y) and a 'Big Number', i.e. a parameter (M) that is always larger than any variable in input to the 'minimum' function. The definition of the shared power is reported for sake of readability:

$$E_{sh}(t) = min\left(P_{PV}(t) + P_{bd}(t) - P_{bc}(t),\, P_{cons}(t)\right) \cdot dt \tag{2.7 revisited}$$

As previously mentioned, the shared power (P_{sh}) is equal to the shared energy (E_{sh}) before being multiplied by the time-step. Considering the definition of the latter and the constraints already set on the various powers involved in the latter, these are certainly always smaller than the quantity $\max(P_{f,max} + P_{bd,max},\, P_{p,max})$. Anyway, when calculating M, this quantity is multiplied by 100, to be even safer:

$$M = 100 \times \max(P_{f,max} + P_{bd,max},\, P_{p,max}) \tag{2.21}$$

The shared power is then *constrained* to be the minimum between the total injections into/withdrawals from the grid, using a set of equations. First, the shared power is constrained to be smaller than both the total injections (2.22b) and the total withdrawals (2.22a):

$$\begin{aligned} P_{sh}(t) \leq P_{PV}(t) + P_{bd}(t) - P_{bc}(t), \\ \forall\, t = 0 : \mathbf{dt\_aggr} : \mathbf{time} \end{aligned} \tag{2.22a}$$

$$\begin{aligned} P_{sh}(t) \leq P_{cons}(t), \\ \forall\, t = 0 : \mathbf{dt\_aggr} : \mathbf{time} \end{aligned} \tag{2.22b}$$

The definition of the auxiliary variable y is also to be *imposed*, using two constraints, so that the variable is equal to 0, when the injections are larger than the withdrawals (2.23a), and 1 otherwise (2.23b):

$$\begin{aligned} [P_{cons}(t)] - [P_{PV}(t) + P_{bd}(t) - P_{bc}(t)] \leq M \cdot y(t), \\ \forall\, t = 0 : \mathbf{dt\_aggr} : \mathbf{time} \end{aligned} \tag{2.23a}$$

$$[P_{PV}(t) + P_{bd}(t) - P_{bc}(t)] - [P_{cons}(t)] \leq M \cdot (1 - y(t)) \,, \quad \forall\, t = 0 : \mathbf{dt\_aggr} : \mathbf{time} \tag{2.23b}$$

The auxiliary variable y can finally be used to constrain the shared power to be larger or equal than the minimum between the injections and the withdrawals:

$$P_{sh}(t) \geq [P_{PV}(t) + P_{bd}(t) - P_{bc}(t)] - M \cdot (1 - y(t)) \,, \quad \forall\, t = 0 : \mathbf{dt\_aggr} : \mathbf{time} \tag{2.24a}$$

$$P_{sh}(t) \geq [P_{cons}(t)] - M \cdot y(t), \quad \forall\, t = 0 : \mathbf{dt\_aggr} : \mathbf{time} \tag{2.24b}$$

Being the shared power already constrained to be smaller than both the injections and the withdrawals, the equations in (2.24) constrain it to be *equal* to the minimum between the two.

The optimisation problem defined by the equations that have just been described is modelled and solved using the environment `pulp`. This is done in the method `battery_optimisation`, from the same-named module, which hence defines the battery's usage strategy during one typical day. The method receives the consumption and production profiles (both in kW), with a time-step of **dt_aggr**, as well as the information about the sizes of the photovoltaic installation and the battery (together with the battery specifications) and the grid contractual powers, which are contained in **technologies_dict**.

The creation (and solution) of the problem using `pulp` is quite straightforward once that the equations to be implemented are defined, therefore this process is only briefly discussed. First, an optimisation problem is created (**opt_problem**), using `LpProblem`. Afterwards, a for-loop is used to iterate along the time-steps, and

to declare, for each one of them, the decision variables (powers and *states*) using `LpVariable`. As previously mentioned, the *states* are logic variables, therefore their category (Binary) is to be specified. On the contrary, powers are real variables, whose lower bound is 0 (positive definite). Similarly, the constraints represented by the equations from (2.10) to (2.24) are added to the problem, iterating along the time-steps[11].

Lastly, the objective to be reached is added to the problem, in terms of objective function. At the moment two different objectives are available: maximising the shared energy (i.e. the sum of **shared_power**, over all the time-steps) or minimising the interactions with the grid (i.e. the sum between **grid_feed** and **grid_purchase**, again over all the time-steps). The first one is set by default and if the user wants to change it, this has to be done manually, from the module. It should be noticed that selecting a different objective in the code also changes the type of problem that is declared (which can be either 'LpMaximize' or 'LpMinimize').

After solving the problem, the method returns the optimised power flows (grid feed/purchase, battery charge/discharge, shared power), the energy stored in the battery at each time-step and the optimisation status of the problem.

It is relevant to notice that, in the code, the number of variables and constraints in the optimisation problem is in direct proportion to the number of time-steps in one single day. Therefore, a 'simple' operation such as choosing a finer time-step (for instance, 15 min), which is fully possible in the tool, would increase the 'size' of the

[11]The command `LpProblem +=`, followed by the expression of the constraint, is used to add the latter to the problem.

problem by four times, with respect to the usual time-step of 1 h, thus increasing the computational time required to solve it.

2.5 Output of the routine

To sum up the content of the previous section, after having obtained both the consumption and production profiles during all typical days of interest, different configurations are evaluated for the aggregate of households. Each one of them is characterised by the size of the photovoltaic installation, in $\mathrm{kW_p}$, and the size of the battery, in kWh. The method `battery_optimisation` is used to optimise the power flows realised in the configuration during a single typical day, thus defining the battery's usage strategy. The method `shared_energy_evaluator` is used instead to obtain the monthly values of the energy (shared, fed into/purchased from the grid, charged and discharged into/from the battery), after having optimised the power flows in all typical days. The monthly results returned by the method are further processed to get the Self-Sufficiency Index ($SSI_{\%}$) and the Self-Consumption Index ($SCI_{\%}$). They are useful indicators about the performances of the configuration, which relate the energy that is shared to, respectively, the total consumption and the total production of energy. The two indices are evaluated as follows:

$$SSI_{\%} = \frac{E_{sh}}{E_{cons}} \times 100 \tag{2.25}$$

$$SCI_{\%} = \frac{E_{sh}}{E_{PV}} \times 100 \tag{2.26}$$

Where: E_{sh} is the shared energy, E_{cons} the total consumption and E_{PV} the total production, either in one month or one year.

The results are processed differently, depending on the simulation that has been chosen: 'fixed-size' analysis for both the photovoltaic system and the battery or 'parametric' analysis for at least one of the two.

Parametric analysis It is assumed that the user is interested in preliminary results on which the different configurations' performances can be quickly compared. Therefore, yearly results are provided, for each configuration, in terms of self-sufficiency and self-consumption indices and shared energy. In addition, the yearly energy values for the other 'actors' of the battery optimisation (consumption, production, grid feed and purchase, battery charge/discharge and stored energy) are provided.

Fixed-size analysis The configuration is analysed in detail, therefore the aforementioned indices and energy values are provided for each month of the year. As previously mentioned, the power flows optimised during each typical days are also shown.

In both cases, the results are printed on the screen in a tabulate format. In addition, CSV files and figures are created and saved in the *'Output/'* folder, respectively, in the sub-folders *'Files/'* and *'Figures/'*. These are discussed in detail in 3, where some simulations are performed. The whole sequence of operations performed during one simulation is summarised in Figure 2.22. These are done in the module `main.py` where the various methods previously described are called, in sequence, to simulate and evaluate the performances of the configuration(s).

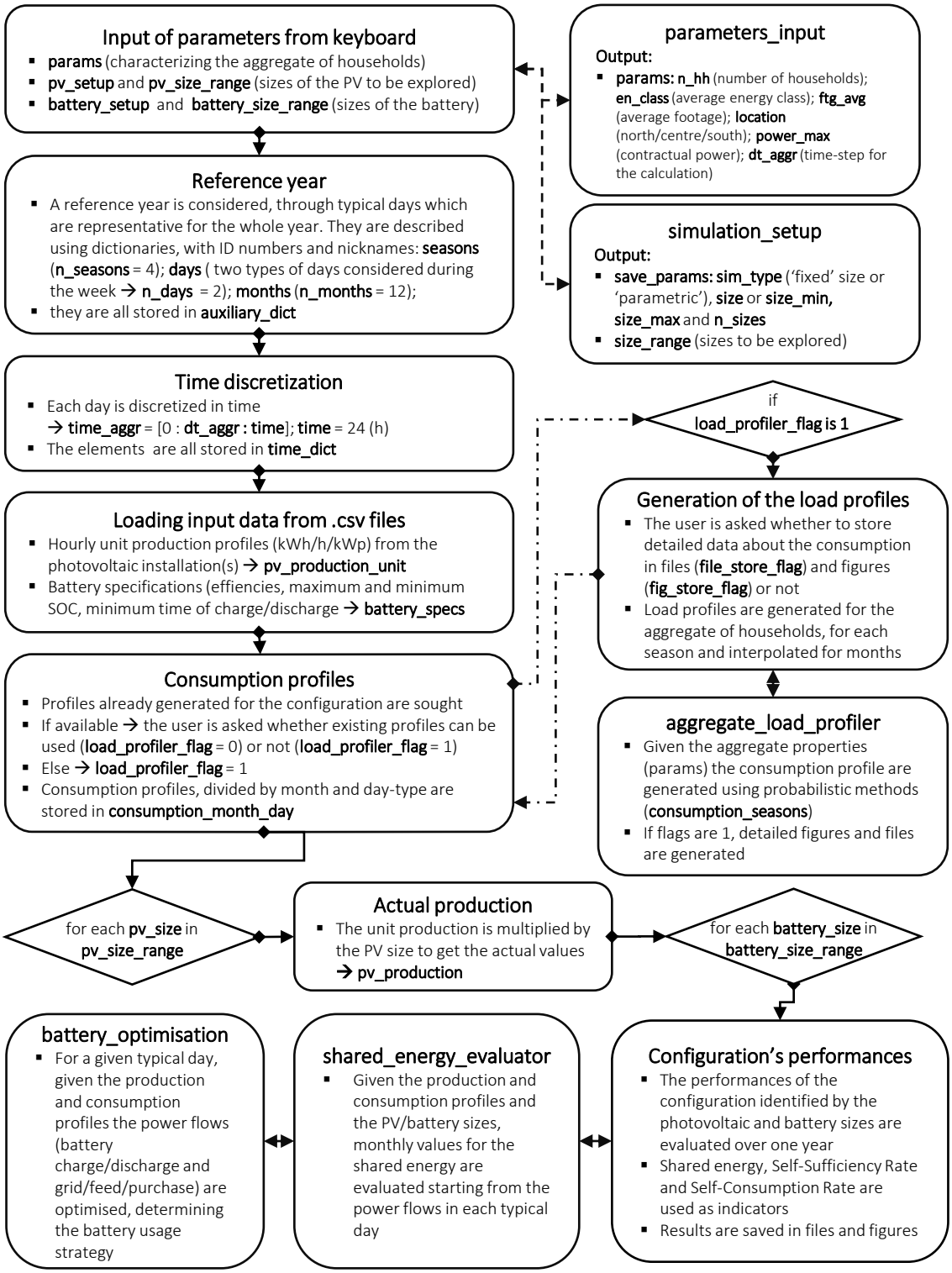


Figure 2.22: *Sequence of all operation performed in the module* `main.py`*. [Self-processing]*

Chapter 3.

Simulation results and discussion

The performances of a potential configuration where locally-produced renewable energy is shared among an aggregate of households have been evaluated using the open-source tool described in 2. The objective of the chapter is, however, to show and discuss the results provided by the tool rather than to evaluate an actual case-study. Therefore, the focus is on some interesting findings that have been deduced from the simulations of the configuration. The results are also used to show the capabilities of the tool, in terms of providing quick and easy-to-grasp evaluations. In all the simulations (whose results are shown and discussed in the following), an energy community composed by an aggregate of 20 households in the North of Italy has been considered. The production profiles previously evaluated using the location of the Politecnico di Torino (shown in Figure 2.7) have been used, while the consumption profiles have been of course simulated by the tool. For this purpose, an average energy class A has been chosen for the appliances so that, as shown later on, the yearly electricity consumption of the households is the same as the Italian average. Furthermore, in compliance with the provisions adopted in the Decree-Law 162/2019, an hourly time-step has been used for the calculation of the *shared energy*. As to the simulations, first a parametric analysis has been performed, varying both the sizes of the photovoltaic system and the battery. Afterwards, some of the combinations between the latter have been simulated again using a fixed-size analysis to provide detailed results about the optimised power flows. From the

latter, it is also possible to better understand the definition of *shared energy* used in the routine.

3.1 Input phase

When running the code, a proper setup for the simulation has to be defined, specifying the values of a number of parameters using inputs from keyboard, as discussed in 2.2.1. The methods used in this phase have been designed with the objective of increasing the flexibility for the user, allowing them to change only the parameters of interest and trying to correct them in case they make some mistakes in typing the expressions required to update the parameters' values. The code's output during this phase is provided in Listing 3.1. The simulation has been performed assuming that the tool was used for the first time, therefore the files where the parameters are saved were not present in the local computer yet and default values applied. The inputs from keyboard are highlighted in red and it can be noticed that some mistakes have been made on purpose, to show the program's ability to correct them.

```
The parameters for the simulation are currently set as follows

Parameter    Value    Unit of measure    Description
-----------  -------  -----------------  ----------------------------------------
n_hh         2        units              number of households
location     north    /                  location (north - centre - south)
power_max    3.0      kW                 maximum (contractual) power
en_class     A+       /                   energy class of the appliances (A+++ - D)
ftg_avg      100.0    m2                  average footage of the households
dt_aggr      60       min                 time-step for the aggregation

Would you like to change any parameter?
Press 'enter' to avoid or
Enter 'ok' to start changing: ok

Upper/lower cases, underscores, quotation marks can be disregarded.
Press 'enter' to stop at any time.

To change a parameter write the whole expression (ex. n hh = 100): n-hhh ======= '100.,'
Maybe you mean n_hh (rewrite the name to confirm): n hh
```

```
Done: n_hh changed to 100 units.

To change a parameter write the whole expression (ex. n hh = 100): n-hh = twenty
Please, enter an integer value for n_hh: 20
Done: n_hh changed to 20 units.

To change a parameter write the whole expression (ex. n hh = 100): [user presses enter]

The simulation for the PV is currently set as follows

Parameter    Value    Unit of measure    Description
-----------  -------  -----------------  ------------------------------------------------------------
sim_type     fixed    /                  Type of simulation for PV: 'fixed' size or 'parametric'
size         2        kW                 Fixed size for PV

Would you like to change any parameter?
Press 'enter' to avoid or
Enter 'ok' to start changing: ok

Upper/lower cases, underscores, quotation marks can be disregarded.
Press 'enter' to stop at any time.

Write the whole expression (ex. sim type = fixed): size = 20
Done: size changed to 20.0 kW.

Write the whole expression (ex. sim type = fixed): [user presses enter]

The simulation for the battery is currently set as follows

Parameter    Value        Unit of measure    Description
-----------  -----------  -----------------  ------------------------------------------------------
sim_type     parametric   /                  Type of simulation for battery: 'fixed' size or '
     parametric'
size_min     1            kWh                Minimum size of the battery
size_max     5.0          kWh                Maximum size of the battery
n_sizes      5            /                  Number of sizes of the battery to be evaluated

Would you like to change any parameter?
Press 'enter' to avoid or
Enter 'ok' to start changing: ok

Upper/lower cases, underscores, quotation marks can be disregarded.
Press 'enter' to stop at any time.

Write the whole expression (ex. sim type = fixed): size min = 0
Done: size_min changed to 0.0 kWh.

Write the whole expression (ex. sim type = fixed): [size max = 30
Done: size_max changed to 30.0 kWh.

Write the whole expression (ex. sim type = fixed): n sizesss = 3
Maybe you mean n_sizes (rewrite the name to confirm): n sizes
Done: n_sizes changed to 3.0 /.
```

```
Write the whole expression (ex. sim type = fixed): [user presses enter]

The parameters for the simulation and the simulation setup are now set as follows

Simulation parameters
Parameter     Value
------------  -------
n_hh          20
location      north
power_max     3.0
en_class      A+
ftg_avg       100.0
dt_aggr       60

Simulation setup (PV)
Parameter     Value
------------  -------
sim_type      fixed
size          20.0

Simulation setup (battery)
Parameter     Value
------------  ----------
sim_type      parametric
size_min      0.0
size_max      30.0
n_sizes       3
```

Listing 3.1: *Code's output during the parameters input phase (user's inputs from keyboard are highlighted in red).*

Entering the required parameters also means defining the aggregate's characteristics in terms of location, number of households and so on. At this stage the tool usually starts looking in the *'Output/'* folder to check if the consumption profiles for the same aggregate of households have already been simulated previously. It should be recalled indeed that the user is given the possibility to choose whether to use the same consumption profiles as in a previous simulation or to simulate new ones. In case the specific aggregate is being simulated for the first time or the user prefers to simulate the consumption profiles again, the program asks them whether detailed files and figures about the aggregate's consumption are to be generated. In

the case of the simulation in exam, there were no load profiles previously generated since the tool was being used for the first time. Anyway, an example of the code's output during this phase is provided in Listing 3.2.

```
Some load profiles have already been evaluated for the current configuration, do you want to use
      them?
Press 'enter' to skip and re-evaluate the load profiles
Enter 'ok' to use the available ones:no

Evaluation of the load profiles for the aggregate of households.

Do you want to store files with detailed information about the load profiles generated and the
     related energy consumption?
Press 'enter' to skip or
Enter 'ok' to store: ok

Do you want detailed information about the load profiles generated and the related energy
     consumption to be plotted and stored as figures?
Press 'enter' to skip or
Enter 'ok' to plot: ok
```

Listing 3.2: *Code's output during the phase of loading/simulating the consumption profiles (inputs from keyboard are highlighted in red).*

3.2 Load profiles and energy consumption

The simulation of the load profiles is a preliminary step to the evaluation of the performances of the configuration, since it provides the aggregate's consumption profiles (hourly, or with a different time-step) needed for the calculations. Nonetheless, these methods can also provide more detailed data that can be useful both to discuss the functioning of the methods themselves and for the user, to have more insight on the households' consumption. The total load profiles simulated for the aggregate of households according to the parameters specified in 3.1 are shown in the figures from 3.1 to 3.4, for all seasons, from winter to autumn respectively, and for both day-types.

It should be recalled that, apart from the lighting[1], the appliances' consumption depend on the season according to the seasonal coefficients only. The latter are however mainly equal to 1 (see Table 2.6), due to a lack of significant data. It is reasonable to assume, therefore, that the differences in the load profiles in different seasons, simulated for this particular aggregate, are mainly due to the small number of households considered, which increases the randomness in the simulation process. The two summer days represent an exception to this, since there is in general a higher consumption due to the presence of the *continuous* power demand from the air-conditioner, which creates a sort of base-load. A similar observation can be made about the differences between the load profiles in the two day-types, due the presence of a small number of appliances whose *average daily load profile* depends on the type of day in the week, namely the *duty cycle* appliances and the ovens.

The load profiles shown in the following figures have been used to evaluate the performances of the configurations that are described later on.

[1]Different *average daily load profiles* are used for the lighting, according to to the season.

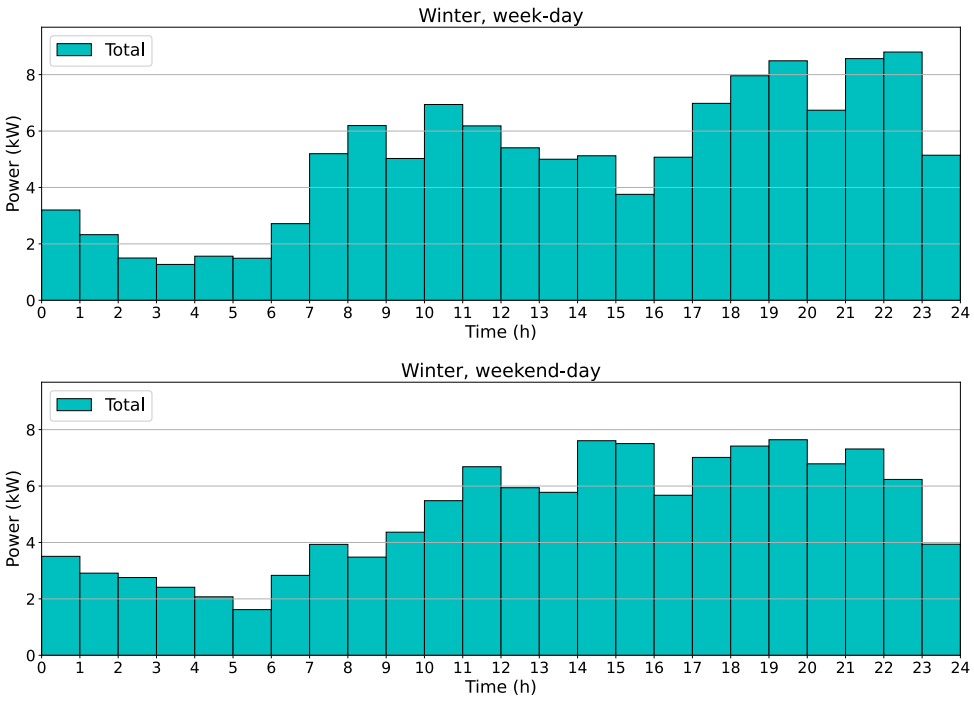


Figure 3.1: *Consumption profiles simulated in* winter *(aggregate of 20 households in the North of Italy, average energy class A). [Self-processing]*

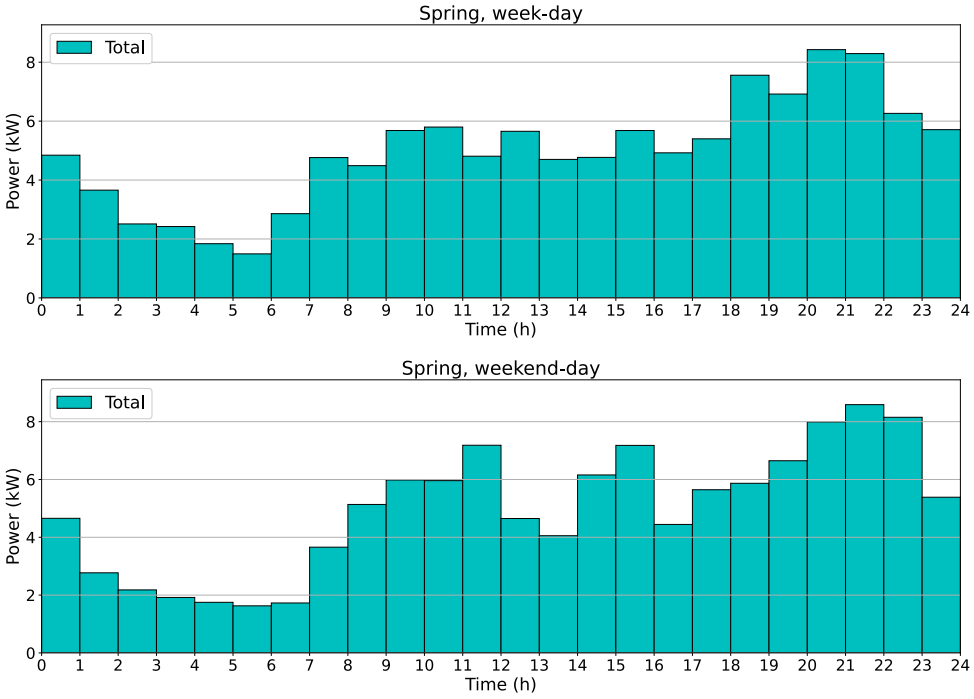


Figure 3.2: *Consumption profiles simulated in* spring *(aggregate of 20 households in the North of Italy, average energy class A). [Self-processing]*

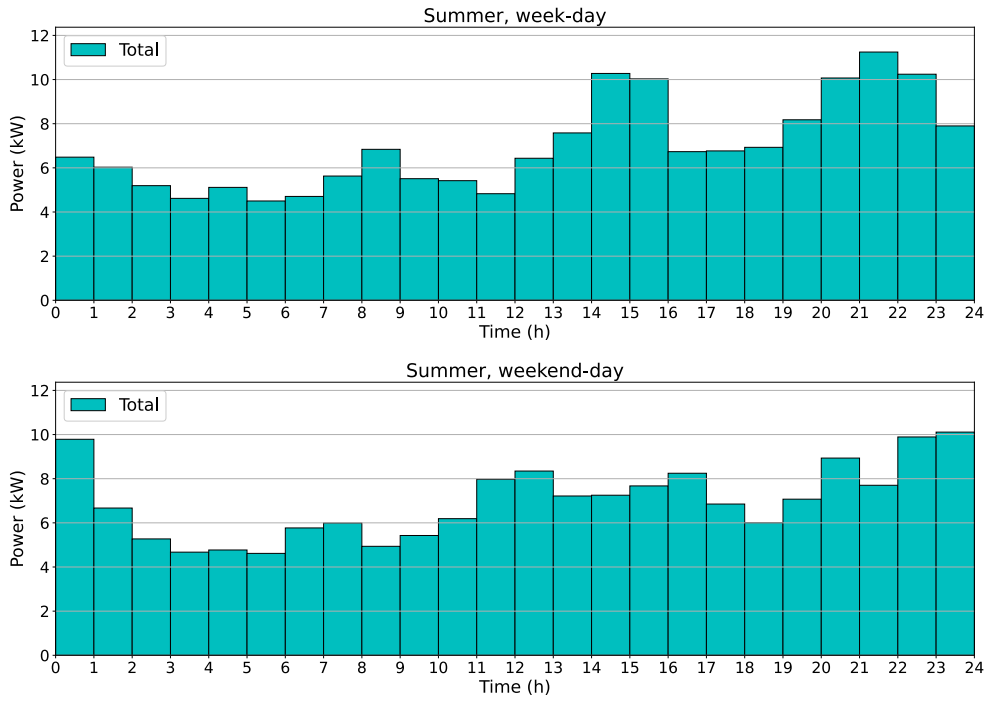


Figure 3.3: *Consumption profiles simulated in* summer *(aggregate of 20 households in the North of Italy, average energy class A). [Self-processing]*

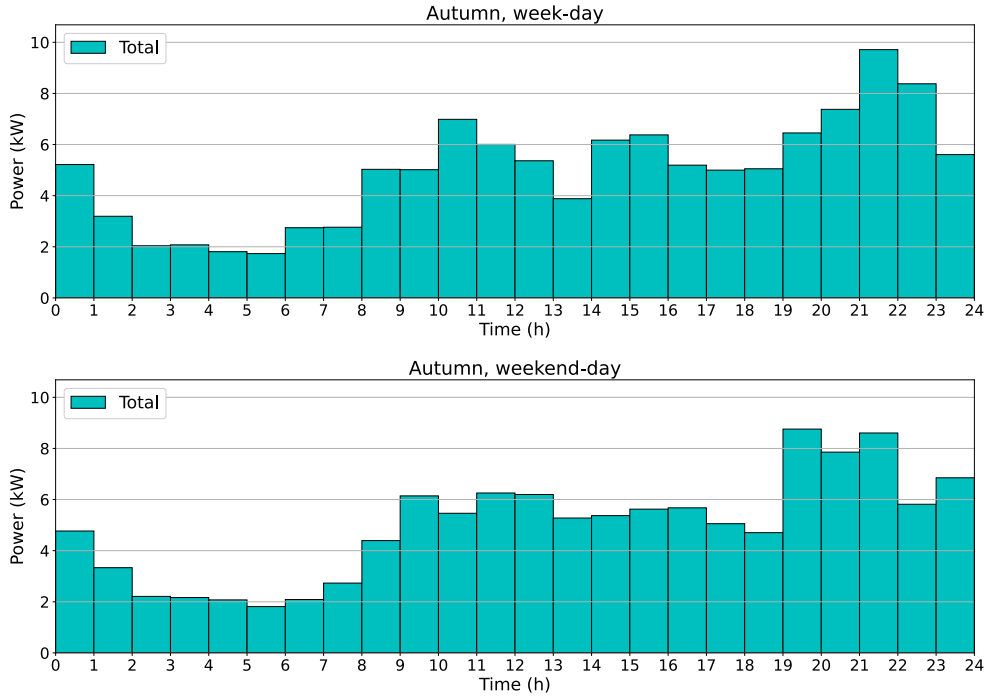


Figure 3.4: *Consumption profiles simulated in* autumn *(aggregate of 20 households in the North of Italy, average energy class A). [Self-processing]*

In general, the shape of the simulated load profiles can be different from the typical normalised profiles for household customers. The latter usually shows indeed two peaks, the first one around noon and the second one during the evening, while the power demand is very small at night. In order to obtain such shape, a large number of households should be considered, as in Figure 3.5 where the profiles generated simulating an aggregate of 2000 households are shown. When a small number of households is simulated, instead, it is more difficult to find such well-defined pattern in the power demand. As to the profiles in Figure 3.5 it can be noticed that, actually, three peaks can be individuated during the summer day (the third one is in the early afternoon). Moreover, the power demand at night is not very small. This is due to the strong influence of the air-conditioner (given its relatively high energy consumption), whose power demand is assumed to be *continuous* throughout the day and has a peak in the afternoon (see Figure 2.14).

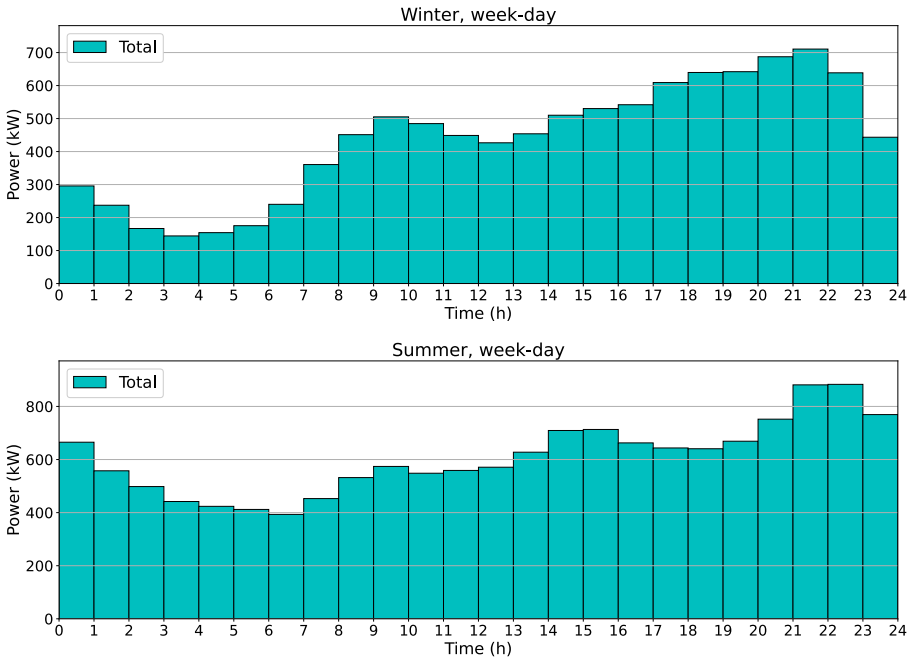


Figure 3.5: *Consumption profiles simulated for an aggregate of 2000 households in the North of Italy, average energy class A+. [Self-processing]*

Another relevant aspect about the simulation of the load profiles is that, despite being different from the normalised profiles, the peaks in the total power demanded from the aggregate of households are relatively small if compared to the maximum available power. The highest peak in the profiles simulated for the aggregate of 20 households is indeed around 11 kW (Figure 3.3). On the other side, the maximum power available for the aggregate, i.e. the contractual power of a single household multiplied by the number of households, is equal to 60 kW. Of course, this does not mean that all households' power demands are simultaneously this far from the maximum available power. This is rather the result of aggregating the households together, whose power demand can instead be, individually, larger or smaller than the average value. There is indeed a quite large variability in the single households' load profiles, as shown in Figure 3.6, where the average load profile in each time-step is compared to the maximum and minimum ones[2].

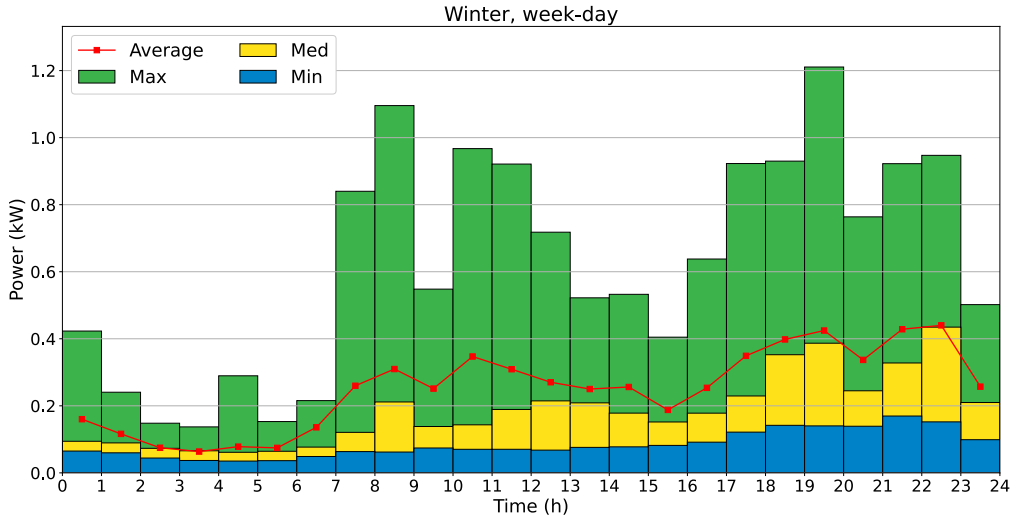


Figure 3.6: *Average, minimum, medium and maximum load profiles simulated for an aggregate of 20 households in the North of Italy, average energy class A. [Self-processing]*

[2]It should be recalled that the maximum and minimum load profiles are considered as quantile of the aggregate of households (respectively, the 90^{th} and 10^{th} household).

Yet, the aggregation of the households is not the only cause in the levelling of the total (and, of course, average) power demand. Changing the time-step of the profiles, indeed, means replacing 'instantaneous' (i.e. in one minute) values with uniform ones, keeping constant the energy consumption in the time interval, thus levelling out peaks and troughs. The 'instantaneous' power demands of the single households are instead very likely to reach rather high values, as in Figure 3.7, where these are shown for a random sample of households.

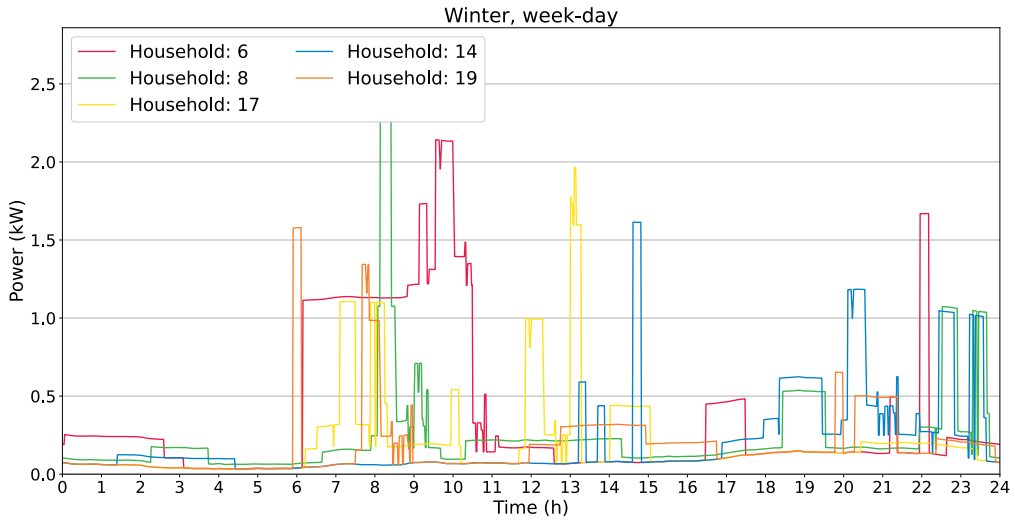


Figure 3.7: *Load profiles of a random sample, simulated for an aggregate of 20 households in the North of Italy, average energy class A. [Self-processing]*

3.2.1 Energy consumption

A detailed analysis of the energy consumption from the electric appliances or the aggregate of households are out of the scope of this work. It has been stressed indeed that the focus is on the profiles of the power demands from the households, which are needed to evaluate the configuration's performances through the optimisation of

the power flows. Nonetheless, it is worth discussing some evidences that arise when considering the yearly and average energy consumption from the appliances.

As to the latter, it can be used to assess if the routine is consistent with the appliances' yearly energy consumption used as inputs for the simulation. The average values in Figure 3.8 are evaluated considering the total units of an appliance found in the aggregate rather than the number of households. Hence, they must be more or less equal to the yearly energy consumption shown in Table 2.5, under the same energy class (A), whose values are reported in the figure (in yellow, above the average values resulting from the simulation), for a better readability.

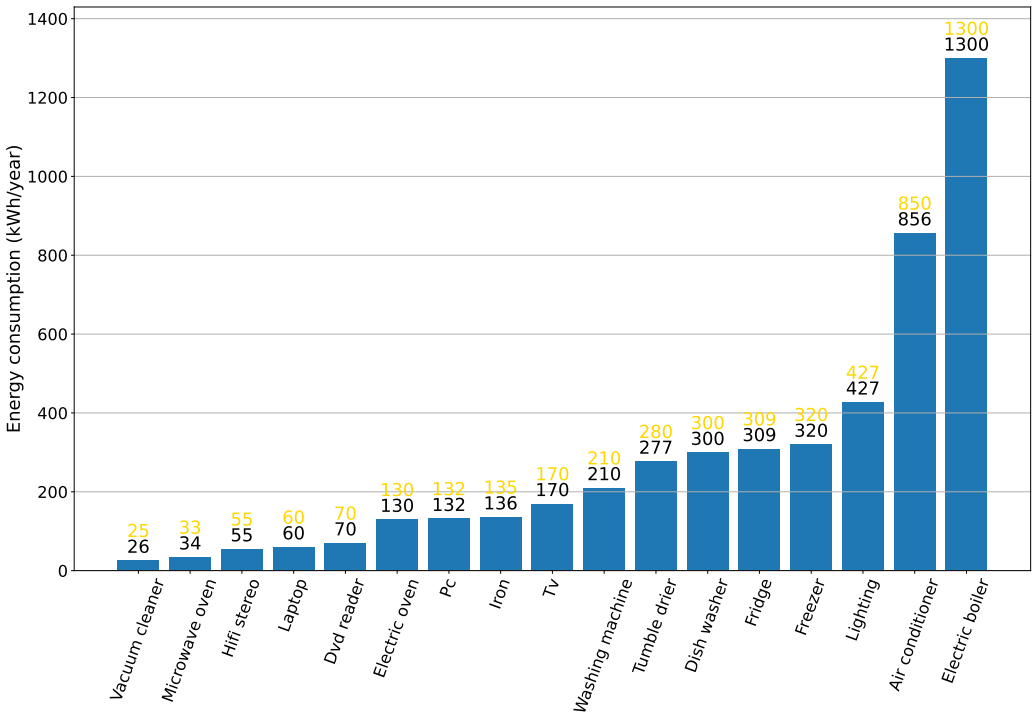


Figure 3.8: *Average yearly energy consumption of the electric appliances under energy class A. [Self-processing]*

The total energy consumption on the other hand is strongly influenced by the appliances' distribution factors. In Figure 3.9 it can be noticed indeed how having

a large yearly energy consumption does not necessarily mean weighing much on the total consumption. An example of this is represented by the electric boiler, which, despite having the largest consumption (in terms of single appliance), it does not have the largest share of the total consumption since it is not widely distributed in the households.

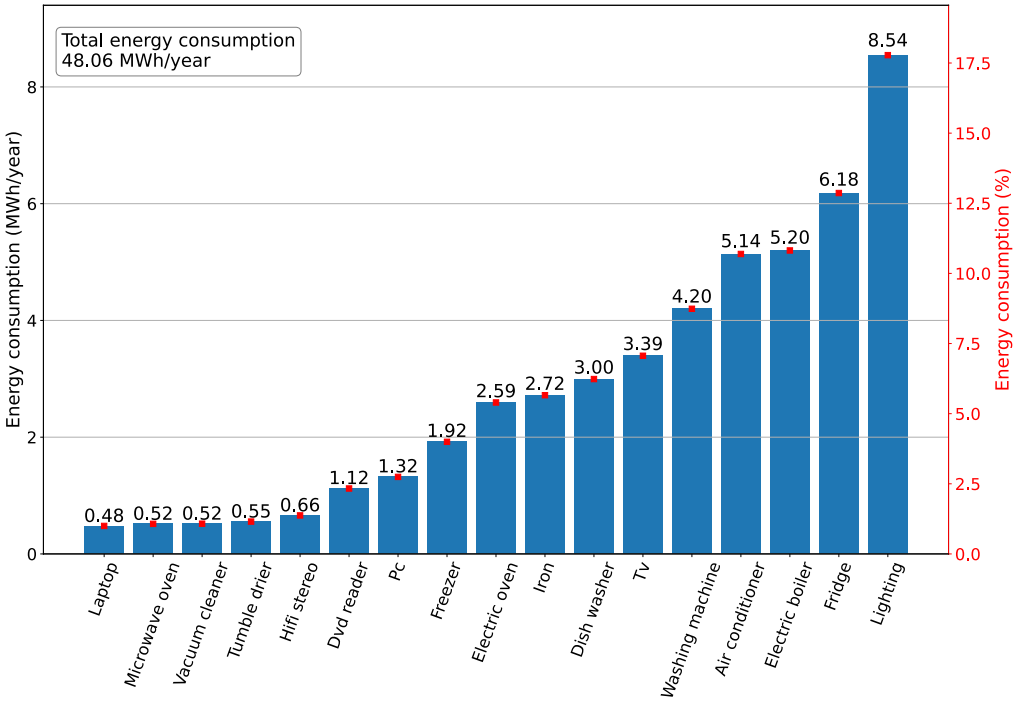


Figure 3.9: *Total yearly energy consumption of the electric appliances under energy class A. [Self-processing]*

From the total consumption shown in Figure 3.9, it can be deduced that the single household's yearly energy consumption, under this energy class, is approximately 2400 kWh/year. The value is equal to the average energy consumption reported in [33] for the Italian households in 2009. Incidentally, this energy class (A) is also in the exact middle between the highest (A+++) and the lowest one (D): as a rule

of thumb, when the households' yearly consumption is larger than the average, a smaller energy class should be selected and vice versa.

The same data broken down by seasons can be used to better grasp how the consumption (hence the load profiles) varies according to the seasonal coefficients. As shown in Figure 3.10, this is more evident for the air-conditioner, which is only used in summer, and less for other appliances, such as the electric boiler or the lighting which are only used less in summer rather than in winter.

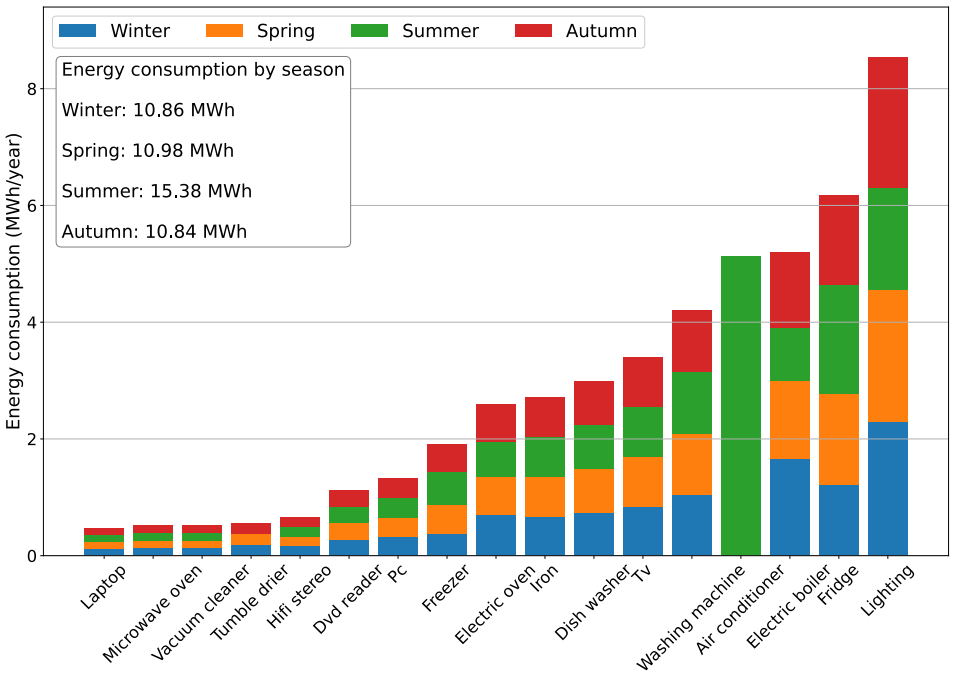


Figure 3.10: *Total yearly energy consumption of the electric appliances under average energy class A, divided by season. [Self-processing]*

Lastly, it should be noticed that only the main appliances that are usually found in the households are considered in the routine, therefore the actual energy consumption might be underestimated. Nonetheless, thanks to the automatisation of the methods used for the simulation of the load profiles, it is quite easy to include

more appliances. If the data needed for the calculation (yearly energy consumption, average daily usage, and so on) are available, indeed, it is enough to add new rows to the CSV files where these quantities are stored. Of course, also the files containing *average daily load profile* and, if needed, the typical work-cycle, are to be included in the *'Input/'* folder, properly formatted. A different situation arises if the standby powers are to be considered. The latter are the average powers absorbed by the appliances during the standby mode, which can be defined as *«the lowest power consumption mode which cannot be switched off (influenced) by the user and that may persist for an indefinite time when an appliance is connected to the main electricity supply»* [35]. In other words, the standby power is the consumption from an appliance when it is plugged in but it is not in an operational mode (i.e. it is not serving its primary function). The standby powers are a relevant issue since they averagely represent 10 % of the yearly consumption in a household [35]. One way to take them into account could be considering a fictitious appliance that provides a base-load throughout the whole day, whose power demand is evaluated in order that its yearly energy consumption is 1/10 of the total. However, this would not take into account that the standby power depends on the appliances that are found in a household (especially the electronic devices) and that the standby mode is complementary to the active mode of these appliances.

3.3 Evaluation of the configurations

The optimisation of the power flows into/from the battery and the grid during all the typical days allows to assess the configurations' performances in terms of shared energy and self-sufficiency/self-consumption indices. Anyway, before moving to the discussion of the results provided by the tool, it is worth discussing the reasons

behind the necessity of including storage systems in a configuration where energy is self-consumed or shared through collective self-consumption. For this purpose, a configuration without a storage system has been evaluated. In the following figures, the consumption and production profiles in a week-day in July are compared to the energy shared by the configuration, evaluated considering different sizes of the photovoltaic installation. The *shared energy* is, according to its definition, the minimum during each hourly time-step between the injections into and the withdrawals from the grid, which in this case are, respectively, the whole production and the whole consumption.

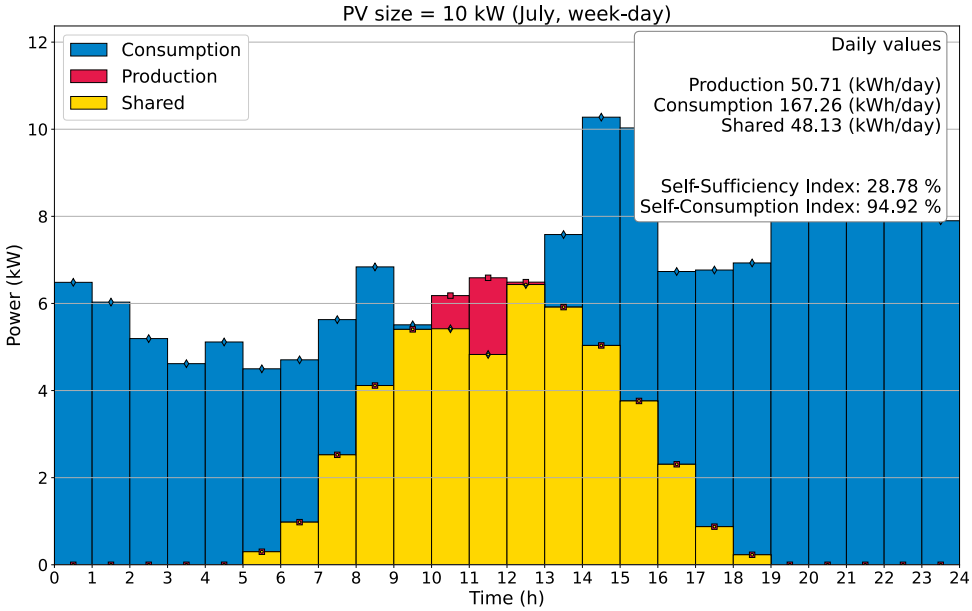


Figure 3.11: *Production, consumption and shared energy in a configuration without storage system (undersized PV system). [Self-processing]*

In Figure 3.11, the results for a photovoltaic system of 10 kW_p are shown. It can be noticed that the installation is rather undersized for the configuration. Almost all the production is shared in fact (the self-consumption index is close to 100 %),

but the shared energy is far from reaching the total consumption. Nonetheless, there is already a small excess in the production, when the latter is larger than the consumption and therefore cannot be shared 'instantaneously'.

In Figure 3.12, the same results are shown for a three times larger photovoltaic size (30 kW_p). Of course, the shared energy and, consequently, the self-sufficiency index increase but in a relatively smaller way (they get approximately 1.6 times larger). The self-consumption index instead decreases dramatically (to around 50 %) since a large part of the 'added' production cannot be 'instantaneously' shared and is therefore in excess.

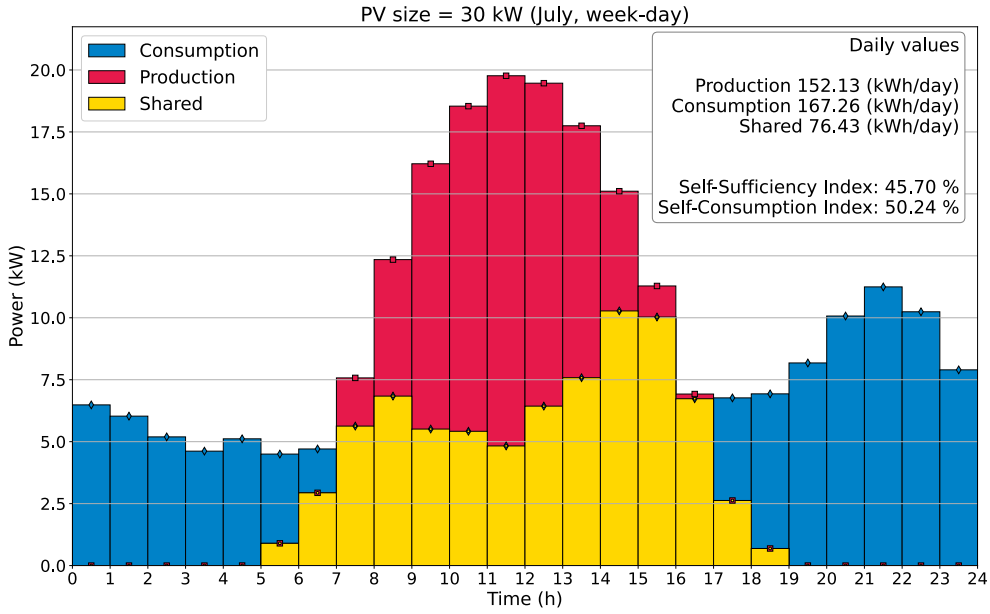


Figure 3.12: *Production, consumption and shared energy in a configuration without storage system (PV system properly sized). [Self-processing]*

In Figure 3.13 (where yearly values are shown), it can be noticed that a further increase in the size of the photovoltaic installation would bring almost no increase in the energy shared within the configuration, even when the total production exceeds

the total consumption. The shared energy shows indeed an asymptotic trend, with a self-sufficiency index that is always smaller than 50 %. On the other hand, the self-consumption index keeps decreasing, meaning that always more production cannot be 'instantaneously' self-consumed.

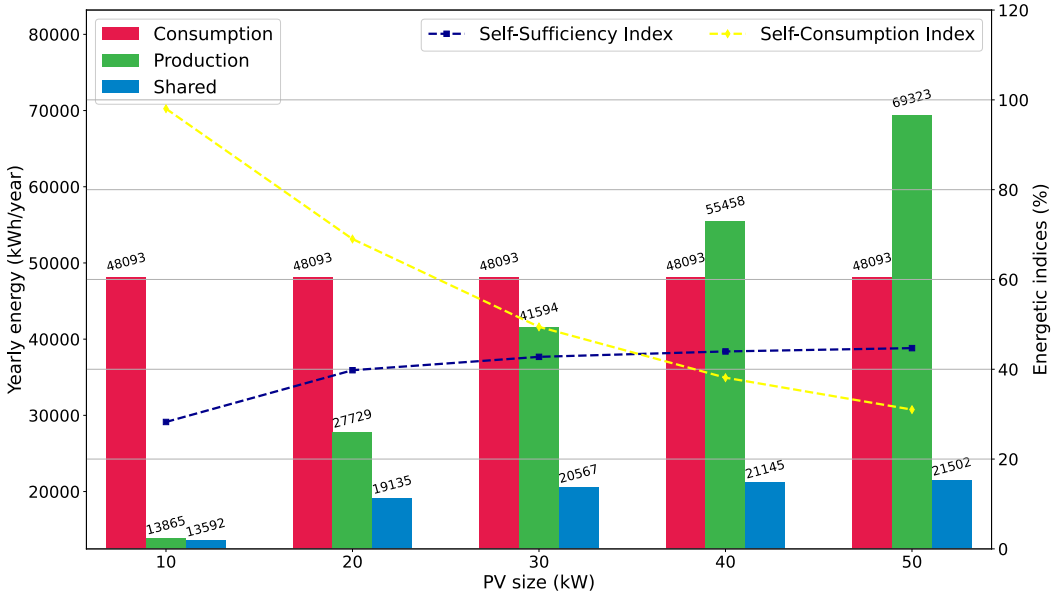


Figure 3.13: *Saturation of the energy shared in a configuration where there is no storage system. [Self-processing]*

Clearly, just increasing the size of the photovoltaic installation does not change the fact that the production from the photovoltaic system is concentrated in the central hours of the day, while the consumption is distributed more uniformly throughout the day. Looking again at Figure 3.12 indeed, it can be noticed that the total production from a photovoltaic system of 30 kW_p (during the typical day, not for the whole year) was already large enough to almost fulfill the consumption, in terms of daily energy (top-right corner of the figure). However, since the production profiles cannot be changed, the only way to try to make them match the consumption is act-

ing on the latter[3]. As previously mentioned, the households' load profiles are taken 'as they are', since techniques involving the change in the customers consumption habits are not considered. The solution adopted is instead shifting the consumption of energy through the use of a battery. Therefore, the production is stored when there is an excess and used later on during the day. Conceptually, this can be seen as distributing the red area in Figure 3.12 (excess) over the blue area (deficit).

3.3.1 Parametric analysis

Having established the need for a storage system to overcome the different distribution in the day of the production and consumption of energy, the relative influence of the sizes of the photovoltaic installation and the battery on the configuration's performances can be addressed. For this purpose, the tool has been used to perform a 'parametric' analysis both on of them. Hence, keeping constant the characteristics of the aggregate (and the consumption profiles previously simulated), a number of configurations has been simulated, which are identified by the coupling between:

- A photovoltaic system's size, which can either be 15, 20, 25, 30 or 35 kW_p;
- A battery's size, which can either be 0, 10, 20, 30 or 40 kWh.

The objective of the optimisation has been set initially to the minimisation of the interactions with the grid. The results of the parametric analysis are shown in Figure 3.14, where the configurations are placed in a space identified by their yearly

[3]It should be noticed that 'consumption', which has always been used as a synonyms of 'load' so far, is here intended as the energy that is used by the households, either to fulfill the power demand from the electric loads or to store energy in the battery. However, in the following the two terms are used again as synonyms, since the energy that flows in the battery is addressed as 'battery charge'.

self-consumption and self-sufficiency indices. A chart of such type is useful when varying simultaneously the sizes of both the photovoltaic system and the battery, since it allows to easily assess how the configuration's performances are affected by each one or both of them. Ideally, the aim is to have a configuration where all the energy that is self-generated is also consumed locally, i.e. shared, and where the shared energy is enough to fulfill the whole energy demand. In other words, this means having the largest self-consumption and self-sufficiency indices as possible. Therefore, configurations that are located in the top-right region of the chart are to be considered as better-performing (from an energetic point of view). Consequently, a variation in the size of the battery or of the photovoltaic system is to be considered more valuable when it shifts the configuration towards this region.

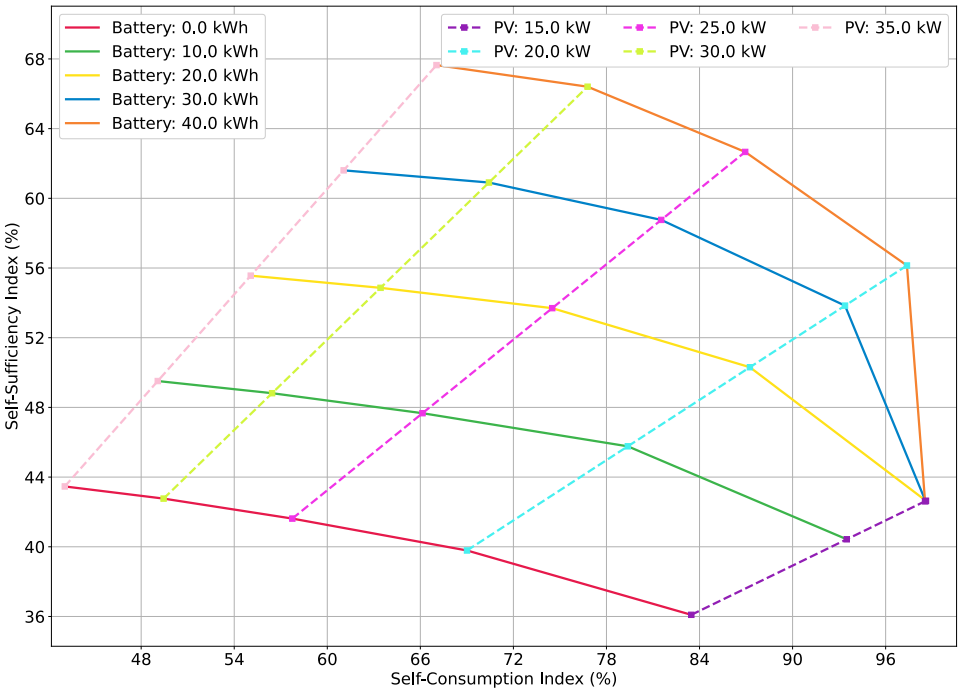
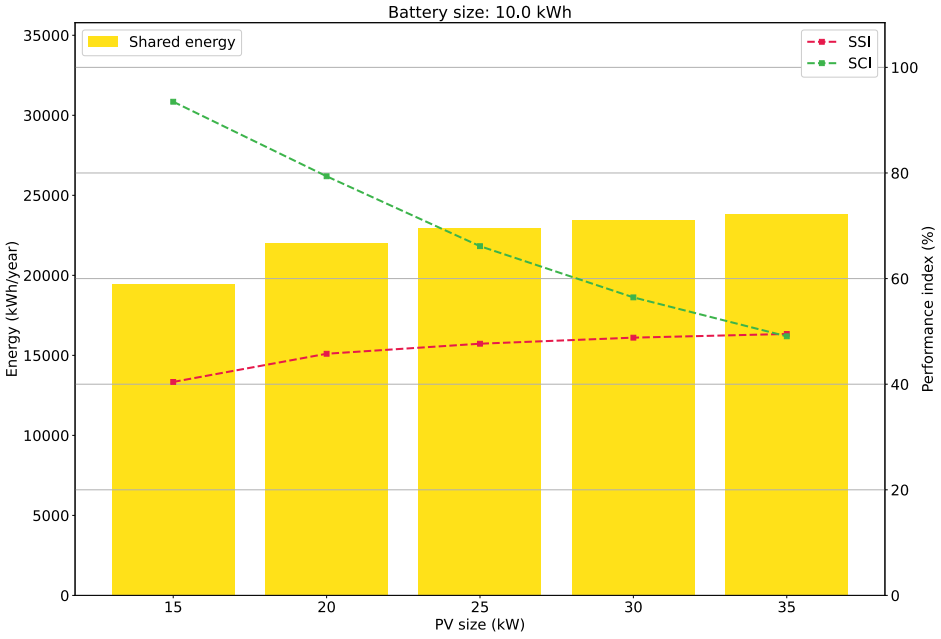


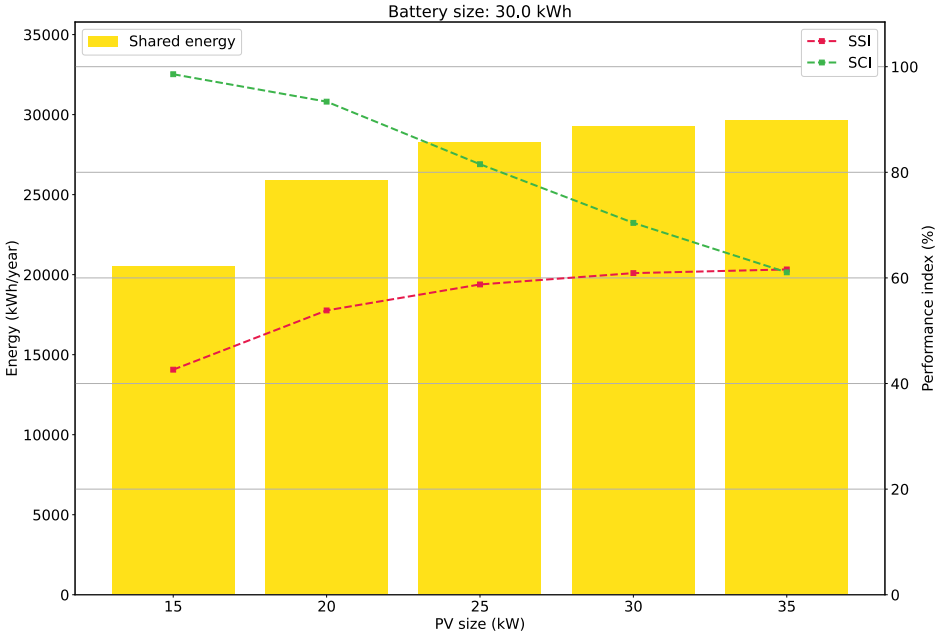
Figure 3.14: *Influence of the photovoltaic system's and battery's sizes on the yearly performance indicators in the minimisation of the exchanges with the grid. [Self-processing]*

As previously mentioned, increasing the size of the photovoltaic system allows to share a larger quantity of energy, thus increasing the configuration's self-sufficiency. Anyway, if the battery's size is not simultaneously increased (continuous lines in the figure), the shared energy increases in a slower way than the production, since an always smaller portion of the latter can be shared hence the self-consumption index decreases. Moreover, the self-sufficiency index does not increase constantly but tends to be saturated. This holds for all configurations, regardless of the presence of a storage system. Anyway, when the battery's size is larger, the increase in the self-sufficiency index is faster and at the same time the decrease in the self-consumption index is slower, thus confirming the important role played by the battery. As to the latter, instead, increasing its size while keeping constant the photovoltaic system's (dashed lines in Figure 3.14), the self-sufficiency and self-consumption indices increase in the same proportion (linear trend). This is true until the maximum self-consumption is reached: moving along the line relating to the photovoltaic system's size of 15 kW_p, it can be noticed that, from that point on, a further increase in the battery's size does not bring any increase in the self-sufficiency, since all the production is shared already.

Interestingly, when the same configurations have been simulated using the maximisation of the shared energy as the optimisation objective, the same results have been obtained, not only in relative terms but also as absolute values. This means that the minimisation of the interactions with the grid also leads to the maximisation of the shared energy, under the definition of the latter used in the routine. The yearly performances of some of these configurations, simulated using the maximisation of the shared energy as the objective of the optimisation, are shown in Figure 3.15 and 3.16. In the latter, either the size of photovoltaic system is varied, while keeping constant the battery's, or vice versa.

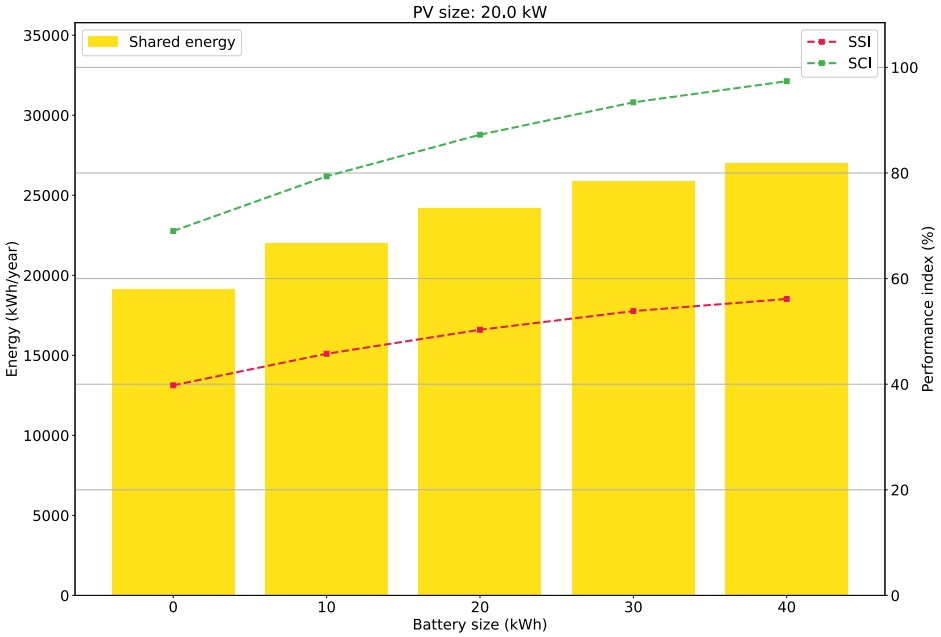


(a)

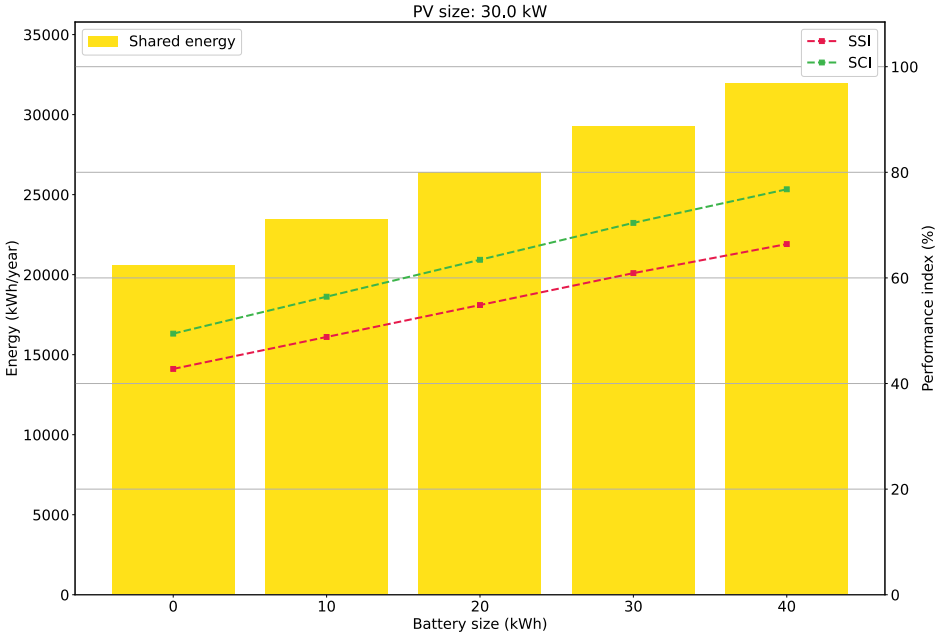


(b)

Figure 3.15: *Variation of the self-consumption and self-sufficiency indices and yearly shared energy with the photovoltaic system's size, for a battery's size of: (a) 10 kWh and (b) 30 kWh. [Self-processing]*



(a)



(b)

Figure 3.16: *Variation of the self-consumption and self-sufficiency indices and yearly shared energy with the battery's size, for a photovoltaic system's size of: (a) 20 kW_p, (b) 30 kW_p. [Self-processing]*

If the single configurations' self-consumption and self-sufficiency indices were transposed into a chart such as the one previously shown, it could be checked that the two optimisation objectives lead to the same results. Moreover, it can be noticed that, while all the previous considerations on the self-consumption and self-sufficiency indices hold, the energy that is yearly shared increases anyway (until eventually being saturated), whether the photovoltaic system's size is increased alone or together with the battery's size. However, this should not shade the fact that increasing the size of the photovoltaic system means a higher capital cost. Hence the full exploitation of the production from the installation should be pursued, which is clearly not the case when the self-consumption index is small.

For sake of completeness, the yearly performances (shared energy and indices of self-sufficiency/self-consumption) of the various configurations simulated are shown in Table 3.1. From the point of view of the absolute values, it is interesting to notice that even when the yearly production from the photovoltaic installation is large enough to fulfill the whole consumption of the aggregate, i.e. for a size of 35 kW_p, the maximum self-sufficiency of the configuration is around 67.5 %. This means that in order to exploit the whole production from the renewable installation, thus fulfilling the whole aggregate's consumption by virtually sharing the former, a larger battery is needed. As a matter of fact, keeping the same battery's size but reducing the photovoltaic system's to 25 kW_p would only slightly decrease the shared energy, therefore the self-sufficiency index, which decreases to around 66.5 %. On the other hand the self-consumption index would increase from 67 % to around 77 %, since the total production is much smaller in proportion, hence the excess that is fed into the grid decreases. As to the configurations with a photovoltaic system of 15 kW_p, it is interesting to notice that when the maximum self-consumption is reached (saturation of the shared energy despite increasing the battery's size), the

self-consumption index is actually around 98 %. This is to be ascribed to the charge, discharge and self-discharge efficiencies of the battery, which cause the energy that can be discharged from it to be slightly smaller than the energy that is charged. For this reason, the energy that is injected into the grid when discharging the battery is smaller than the excess production that is sent to the battery.

Table 3.1: *Comparison between the yearly performances of various configurations (aggregate of 20 households). [Self-processing]*

Configuration		**Yearly energy** (kWh/year)			**Indices** (%)	
PV size (kW_p)	**Battery size** (kWh)	**Production**	**Consumption**	**Shared**	**SSI**	**SCI**
15	**0**	20796.8	48092.6	17359	36.09	83.47
	10			19443.8	40.43	93.49
	20			20505.7	42.64	98.6
	30			20499.5	42.63	98.57
	40			20493.3	42.61	98.54
20	**0**	27729.1	48092.6	19135.2	39.79	69.01
	10			22010.1	45.77	79.38
	20			24193.1	50.31	87.25
	30			25893.6	53.84	93.38
	40			27000.7	56.14	97.37
25	**0**	34661.4	48092.6	20015.7	41.62	57.75
	10			22924.3	47.67	66.14
	20			25823.6	53.7	74.5
	30			28259.7	58.76	81.53
	40			30136.1	62.66	86.94
30	**0**	41593.6	48092.6	20567.1	42.77	49.45
	10			23476.3	48.81	56.44
	20			26384.6	54.86	63.43
	30			29291.1	60.91	70.42
	40			31936.6	66.41	76.78
35	**0**	48525.9	48092.6	20902.6	43.46	43.08
	10			23812	49.51	49.07
	20			26720.6	55.56	55.06
	30			29627.5	61.61	61.05
	40			32532.7	67.65	67.04

3.3.2 Fixed-size analysis to investigate the optimised power flows

The tool has been used to perform a fixed-size analysis, choosing a single size for both the photovoltaic system (20 kW_p) and the battery (30 kWh), in order to get detailed results about configuration and the optimised power flows during each typical. The configuration has been chosen because of the large self-consumption that it allows to reach in the whole year (around 93.5 %) and relatively larger self-sufficiency (around 54 %). In the following, the focus is on two typical days that can be considered more significant to be compared: a winter day (January, week-day) and a summer day (July, weekend-day). The objective of the optimisation has been set to the maximisation of the shared energy but, as previously discussed, the minimisation of the exchanges with the grid would have led to the same results. The power flows optimised in January are shown in Figure 3.17.

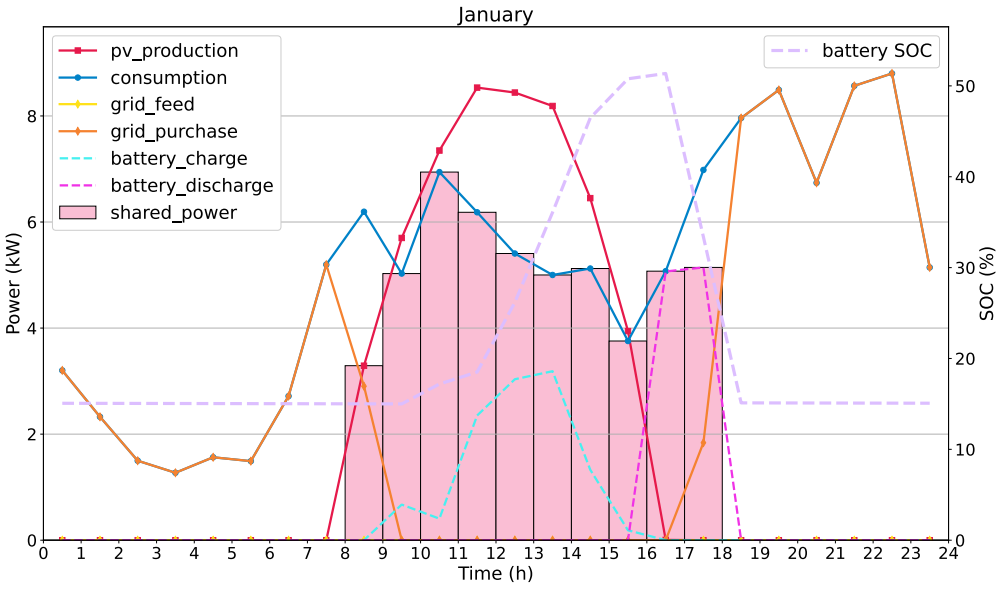


Figure 3.17: *Optimised power flows during a typical day in January (PV system of 20 kW_p, battery of 30 kWh). [Self-processing]*

The production from the photovoltaic system is mainly used to fulfill the demand from the aggregate of households, thus reducing the need to purchase power from the grid. Anyway, when there is an excess in the production, i.e. when it is larger than the consumption, it sent to the battery. Hence, the consumption of the excess production is shifted in time through the use of the battery, where the former is stored in order to be used later on in the day, thus increasing the shared energy. Of course, being the production smaller in winter, the battery's capacity is only partly exploited. Moreover, the stored energy is not enough to fulfill the demand from the aggregate during the whole day.

In order to better understand the definition of the shared energy used in the routine, the latter is reported once again for sake of readability:

$$E_{sh}(t) = min\left(P_{PV}(t) + P_{bd}(t) - P_{bc}(t),\, P_{cons}(t)\right) \cdot dt \qquad \text{(2.7 revisited)}$$

Where: the two terms represent, respectively, the total injections into the grid and the total withdrawals from the grid.

It can be noticed that, using this definition of the shared energy, three situation may arise, also depending on the time of the day:

- In the first and last hours of the day the production is smaller than the consumption and there is no energy stored in the battery;
- During the production hours (central hours of the day), the latter becomes larger than the consumption and the excess is sent to the battery;
- At the end of the production hours, the consumption is again larger than the production but there is energy stored in the battery that can be discharged.

In the first case, the shared energy is equal to the injections into the grid, which clearly coincide with the production (that can also be zero). In the last case, instead, the discharge of the battery allows to make the injections into the grid match the withdrawals (consumption), thus increasing the shared energy. Lastly, in the second case, the shared energy is equal to the withdrawals, since the consumption is smaller than the production. The difference between the two is equal to the energy that is charged into the battery, therefore subtracting the latter from the injections does not affect the shared energy. Nonetheless, considering the latter term in the injections allows to prevent the arise of situations where the optimisation algorithm allows to charge into the battery more energy than the actual excess in the production, thus overestimating the virtual self-consumption (the energy that is shared yearly exceeds the total production). This allows to understand the reason behind the subtraction of the battery charge from the total injections. On the other hand, the mutual-exclusivity between the charge and discharge of the battery ensures that the injections into the grid are not affected by the subtraction the former when discharging the stored energy in the battery. Conclusively, it is clear from the power flows shown in the figure that the definition of the shared energy used in the routine allows to account for the energy stored in the battery just once (more precisely, when it is discharged from the battery).

The power flows optimised in July (for the same configuration) are shown in Figure 3.18, where it can be noticed that the system behaves in a similar way. In this case, however, the production from the photovoltaic system is much larger because a summer month is considered. The size of the battery is not large enough to store the whole excess in the production, therefore some of it is just fed into the grid. As previously discussed, these situations are ineffective because not all the produc-

tion from the renewable installation is shared (not instantaneously nor through the battery) thus decreasing the self-consumption achieved by the configuration.

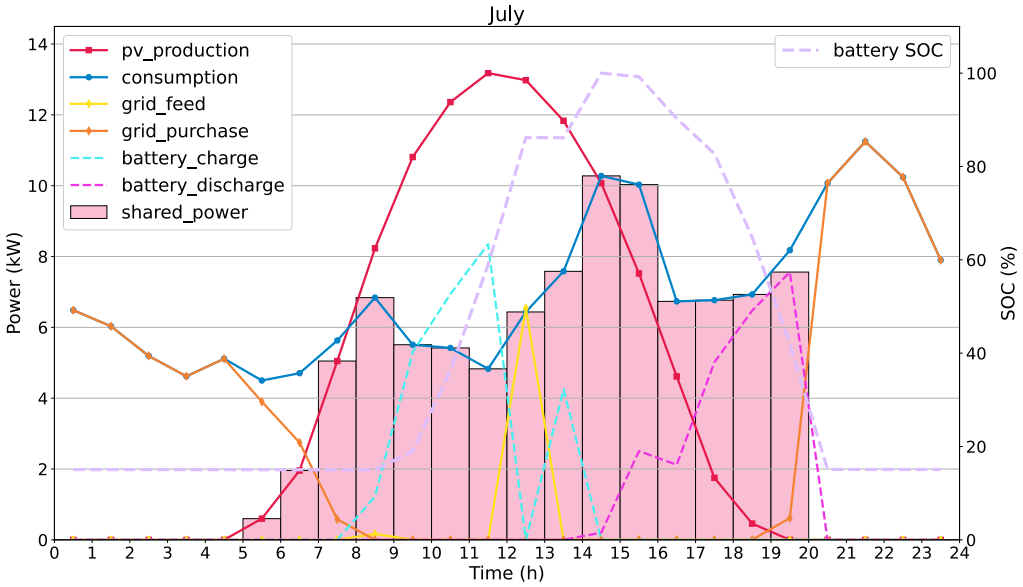


Figure 3.18: *Optimised power flows during a typical day in July (PV system of 20 kW_p, battery of 30 kWh). [Self-processing]*

This is more evident in the power flows shown in Figure 3.19, evaluated for a configuration with a photovoltaic system of 35 kW_p and a battery of 40 kWh. In this case, even in January, when the production from the photovoltaic installation is smaller, the battery is fully charged and a large excess production is fed into the grid. This confirms what had already been concluded with the parametric analysis, i.e. that a larger battery is needed in this case to share and self-consume a larger part of the production. Lastly, in Figure 3.20, the best situation from an energetic point of view is shown. In this case indeed the energy that is shared instantaneously or through the battery is enough to fulfill the demand from the households during the whole day, and almost all the production is shared (the excess production that is fed into the grid is almost zero).

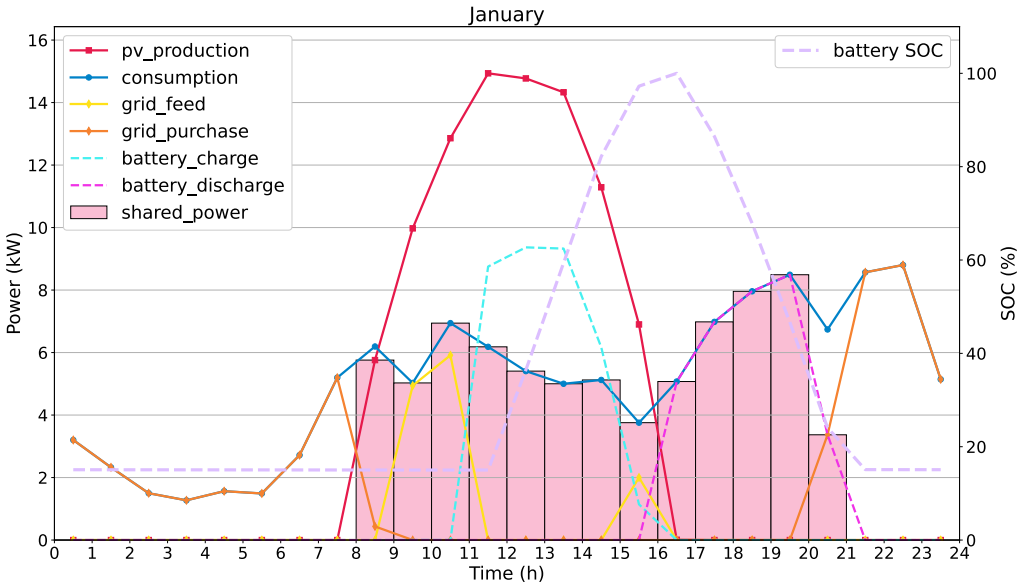


Figure 3.19: *Optimised power flows during a typical day in January (PV system of 35 kW_p, battery of 40 kWh). [Self-processing]*

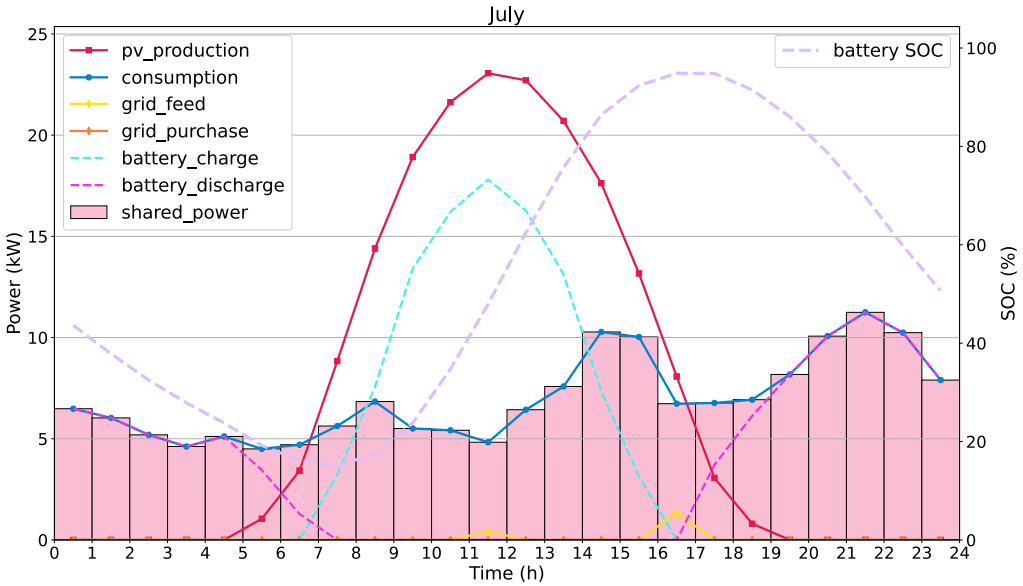


Figure 3.20: *Optimised power flows during a typical day in July (PV system of 35 kW_p, battery of 120 kWh). [Self-processing]*

However, this results has been obtained considering a battery of 120 kWh, which is three times the size considered in previous simulation. Furthermore, the battery is largely oversized for the winter months, where the production is smaller. In

Figure 3.21 indeed, the power flows optimised in January are shown for the same configuration. It can be noticed that only half of the battery's capacity is employed to store the excess production from the renewable installation.

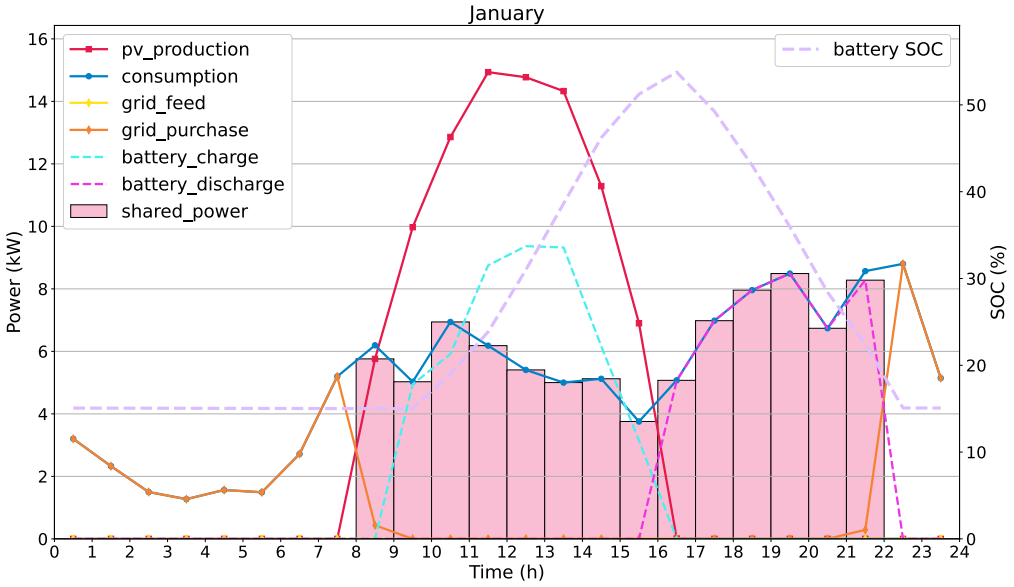


Figure 3.21: *Optimised power flows during a typical day in January (PV system of 35 kW_p, battery of 120 kWh). [Self-processing]*

While this might not be a relevant issue from the point of view of the energetic performances, it gains particular relevance when the economics of the investment are considered. Indeed, the a similar consideration can be made on the battery's size as the one made on the photovoltaic system's: a larger size requires a higher capital cost, therefore the 'full exploitation' of the added capacity should be pursued. In the case of the battery, however, other variables come into play, which can affect its useful life, as briefly discussed in the following.

3.3.3 Further discussion

In Table 3.2 the yearly performances evaluated for two configurations with the same battery's size (30 kWh) and photovoltaic system's size of, respectively, 20 kW_p and 25 kW_p are compared. As mentioned several times, increasing the size of the installation requires a larger capital expenditure but allows to increase the energy shared in the configuration, therefore the self-sufficiency index, and to decrease the energy that has to be purchased from the grid. On the other hand, if the battery's size is kept constant, the self-consumption index is decreased and the excess production that is fed into the grid is increased.

Table 3.2: *Comparison between the yearly performances of two configurations with battery of 30 kWh and PV system of, respectively, 20 kW_p and 25 kW_p. [Self-processing]*

Configuration	**PV size (kW_p)** **Battery size (kWh)**	**20** **30**	**25** **30**	**Δ** **(%)**
Performance indices (%)	**SSI_{year}**	53.84	58.76	+5
	SCI_{year}	93.38	81.53	-12
Energy values (kWh/year)	**Consumption**	48092.6	48092.6	0
	Production	27729.1	34661.4	+25
	Shared	25893.6	28259.7	+9
	Grid feed	1214.6	5648.9	+365
	Grid purchase	22199.0	19832.9	-11
	Battery charge	7379.4	8996.8	+22
	Battery discharge	6758.4	8244.1	+22

The energy that is totally charged into/discharged from the battery over one year is larger because there is a larger excess production to store. However, it is interesting to observe how the usage of the battery changes in relative terms, with respect to the size of the latter. The battery, indeed, can only accommodate a finite amount of energy during its lifetime, which depends on its size. Therefore, the more energy is charged/discharged into the battery the sooner it has to be replaced, thus requiring a new expenditure. A useful indicator of the battery's usage can be

quickly evaluated using the values shown in Table 3.2. In more detail, the values of the energy that is totally charged into/discharged from the battery can be used to evaluate the number of full cycles to which the battery is subject during one year:

$$n_{cycles,\,avg} = \frac{(E_{bc} + E_{bd})_{year}}{2 \cdot CAP_{nom}} \qquad \text{[cycles/year]} \quad (3.1)$$

Where: $(E_{bc} + E_{bd})_{,year}/2$ is the energy stored in average in the battery in one year and CAP_{nom} is its nominal capacity (size), ideally, the energy stored in a full cycle.

For the two configurations shown in Table 3.2, the latter are roughly equal to 235 and to 285 cycles/year, respectively. These quantities can be used together with the average number of cycles to which a lithium-ion battery can be subject during its lifetime, which usually range between 2000 and 10000 [29], to roughly estimate the useful life of the battery. Assuming an average value of 5000 cycles in the battery's lifetime, the latter would last around 21 years in the first case, 17 in the second one (the value are rounded down to the closest integer value). When performing an economic analysis, the duration of the investment is usually set to 20 years, considering both the usual lifetime of a photovoltaic installation and the duration of the incentives on the shared energy. In the first case, therefore, the battery appears to be properly sized in terms of useful life. On the other hand, in the second case, the battery is to be changed before the end of the investment, thus requiring another capital expenditure. It is interesting to notice that, as a consequence of this, even though the smaller energy that is shared (therefore the smaller revenues), the first configuration may results more economically advantageous because of the single capital expenditure that is required on the battery (in the conventional duration of the investment).

The latter aspect allows to better detail an important topic of discussion that has only been *scraped in the surface* so far, i.e. that a full analysis of the configurations requires also an economical evaluation of the investment. Indeed, reaching high self-consumption and self-sufficiency indices is surely captivating, since it means that the renewable production is fully consumed and at the same time the shared energy is quite large if compared to the demand of the aggregate. Furthermore, the higher the energy that is shared, the higher the revenues for the members of the configuration. Nonetheless, the larger the sizes of the photovoltaic system and the battery, the higher the investment (in terms of capital expenditure) that is required. Moreover, as proved by the case of the battery's lifetime, other variables can come into play, which are not always straightforward to consider. Therefore, it can be concluded that when actually designing the configuration (in terms of sizes of the systems) a trade-off between energetic performance and economics is to be found. Fortunately, the tool provides all the energy quantities needed by the user for this task (yearly shared energy, production, battery charge and discharge, and so on). The latter are exported from the code and stored in CSV files in the '*Output/*' folder. Hence, they can be easily accessed by the user and used to perform the most suited economic/financial analyses of the configuration.

Computational times

As mentioned several times, the tool provides quick results about the performances of the configurations that are simulated. The computational time required by the tool to perform the calculation depends on a series of factors. The influence of the time-step on the 'size' of the optimisation problem (therefore, on the computational time) has been briefly discussed already. Another relevant factor is of course the number of households that are considered. In Figure 3.22 the time required to simulate one

configuration is compared for aggregates of increasing number of households (for each configuration, the sizes of the photovoltaic system and the battery have been adapted to the number of households).

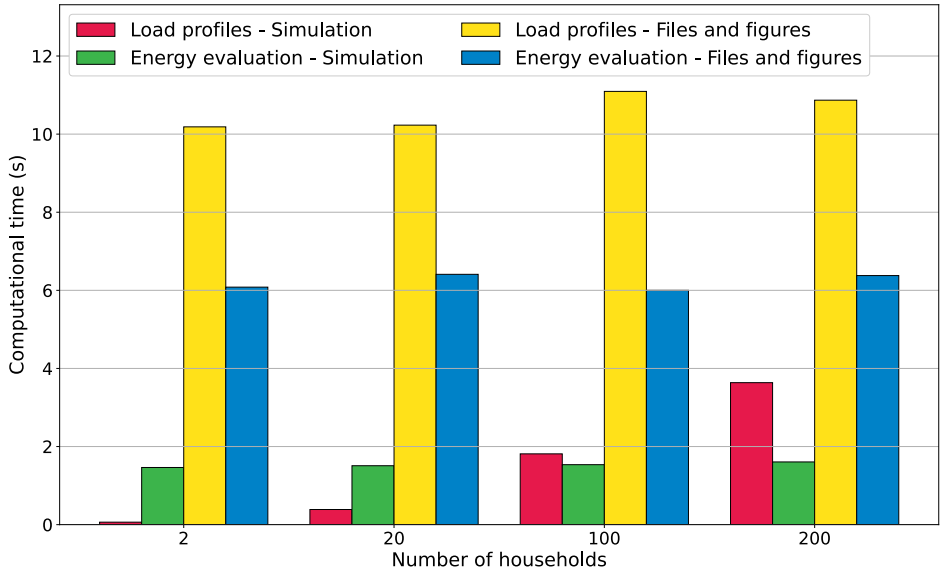


Figure 3.22: *Influence of the number of households on the computational times of the tool (fixed-size analysis). [Self-processing]*

The number of households strongly affects the computational time required to simulate the load profiles. On the other hand, the time required for the evaluation of the configurations is basically constant and does not show a correlation with the number of households. This is due to the fact that, after the aggregation of the household's load profiles, the size of the aggregate does not affect anymore the calculation. In general, it is interesting to notice that the total time for the evaluation of one configuration ranges around 18 and 22 seconds, of which the creation of files and figures represents a relevant share (around 70 and 90 %). Interestingly, the computational time required to optimise and evaluate a single configuration does

not changes with the sizes of the photovoltaic system and the battery that are considered, as shown in Figure 3.23. The only exception is represented by those cases in which no battery is considered, therefore the optimisation is slightly quicker.

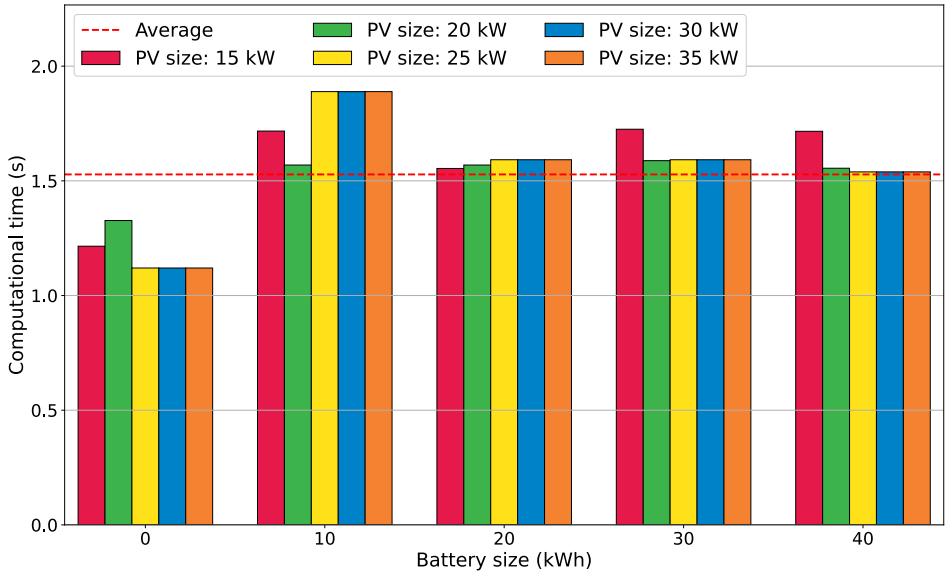


Figure 3.23: *Computational time required to optimise and evaluate different configurations (20 households in the North of Italy, average energy class A). [Self-processing]*

In the cases considered in the figure, the tool took in average 1.5 seconds to optimise and evaluate each configuration. The total time required to simulate the 25 configurations was around 45 seconds (also considering the simulation of the load profile and the creation of files and figures).

Chapter 4.

Sharing the tool via GitHub

So far in this thesis the word 'community' (or declined as 'communities') has appeared about 200 times, 'share' (or 'shared', 'sharing' and so on) about 150 times[1]. Of course, they were referred to a specific practice of sharing energy within a community of customers, which has hopefully been comprehensively described. However, the idea of adding value to something just by sharing it can easily be abstracted to a more general meaning. Taking the tool developed during this work as an example, it certainly could have been created just for a private use, restricted to this particular context. Anyway, chances were high that it would have *fallen into oblivion* within a relatively short time or, best case scenario, it would have been used by other fellow students for further elaboration in the future. Furthermore, quoting the Free Software Foundation: *«if you write a program and use it privately* [...] *you do miss an opportunity to do good»* [43]. The sentence is actually extrapolated from a different context, where the attention was on the dichotomy between free and non-free software, rather than public/private. It further stated in fact that, despite missing the chance to *«do good»*, developing a program for a private use is still better than developing a program that is not free. This helps to introduce the reasons that have driven the choice for open-source.

[1]This result comes from a quick research of these words, using a pdf file reader.

In order to overcome the double meaning of the term 'free' in English (free can either mean 'that has the feature of freedom' or 'not paid for'), the term 'for-free' is used in the following when referring to the monetary aspect. A non-free, or *proprietary*, software is not necessarily not-for-free and similarly a free software is not necessarily for-free. A software is free in fact when it respects the *user's freedoms* to run, study and modify its code and to redistribute copies of it, with or without modifications, once that they have obtained a copy of it [43].

The objective adopted from the beginning was to reach a large number of people, or at least try to. Sharing the tool for-free can help reaching a good number of users who can test it and then provide a feedback about its capabilities. Sharing it as an open-source tool, though, opens up to a much wider range of possibilities. Open-source means, in the first place, that the code is available to whoever is interested in operating on it. There is a series of *practical* advantages arising when making something open-source. For example, it is likely that the code, despite working properly as it has been proved, does not work in the best (i.e. the fastest, most efficient,...) way. In the 'open-source community' there are many experienced programmers who, if interested in the work, can offer their skills to improve the functioning of the code. There is certainly more: even assuming the code already working at its best for what it does, it could still be doing much more. Making the tool's code available to others allows it to be further developed or, for instance, to be included in wider projects. Of course this comes at a price: when writing open-source code in fact, it should be taken into account that other people, with different ways of thinking, will work on it, therefore a more 'objective' point of view should be embraced. Also, every operation should be done in the clearest way and properly commented.

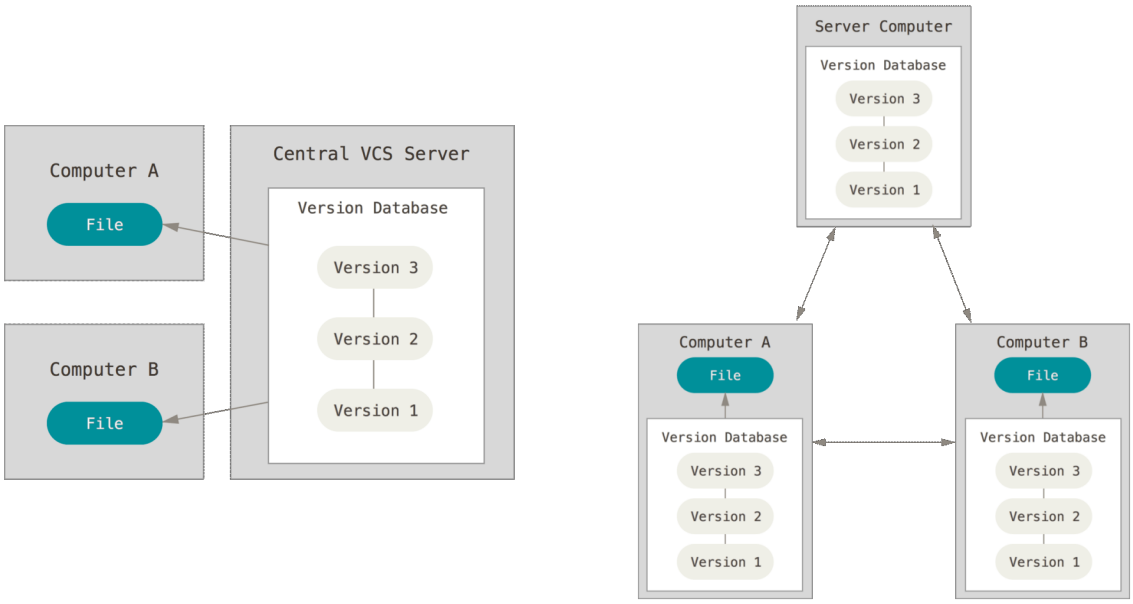
The terms 'free' and 'open-source' have been used as synonyms so far, but they actually derive from two different schools of thought. In very practical terms, almost every program that can be considered free is also open-source and vice-versa [43]. Anyway, on a more abstract level, the 'free software philosophy' goes beyond the aforementioned practical advantages of open-source and sees the *«freedom of the users over the software»* as the primary goal and the basic instinct of programming and as an ideal to always pursue [43]. Of course, this is to be reflected in a number of relevant aspects, such as the choice of a proper licence and the need for a free documentation. The reader can find further information about the 'free software philosophy' at [43].

Having briefly introduced the concepts of free and open-source programs, some more practical aspects of making the tool open-source can be addressed. In the rest of the chapter the development platform GitHub, which is the largest in the world [44], is briefly described together with the distributed version control system Git, that has been used (and is going to be used in the future) for the development of the tool. Hence, the basic functioning of both are introduced in order to help potential contributors.

4.1 Using Git to manage the code's development

Version control systems (VCSs) are useful tools in managing the development of a project which consists of a series of code files (but also of other kinds of project), keeping track of the changes done on the files through a *version database*. A version is a *«snapshot»* of the status of the files in a specific moment [45]. Unlike centralised version control systems, which rely on a central server where the *version database* is stored and from which local computers (clients) can look at the files in a specific

version, in a distributed system, such as Git, each client looks at the whole database at once [45]. Both centralised and distributed systems allow for group-working on a project. Anyway, having all versions available locally can overcome some serious problems arising in case of a failure of the central server [45]. A comparison between the two types of system is shown in Figure 4.1.

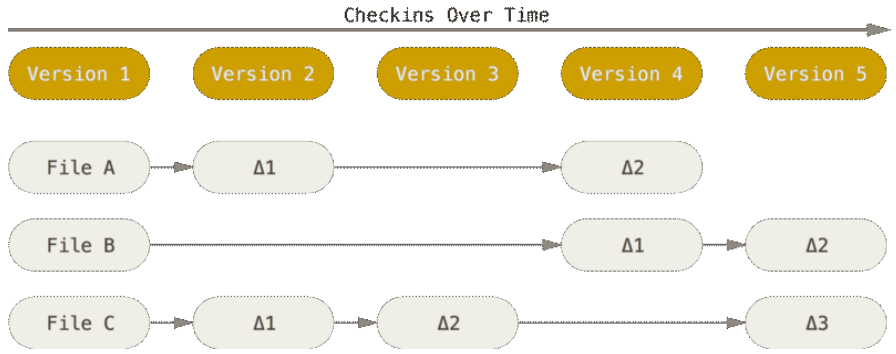


(a) Centralised VCS *(b) Distributed VCS*

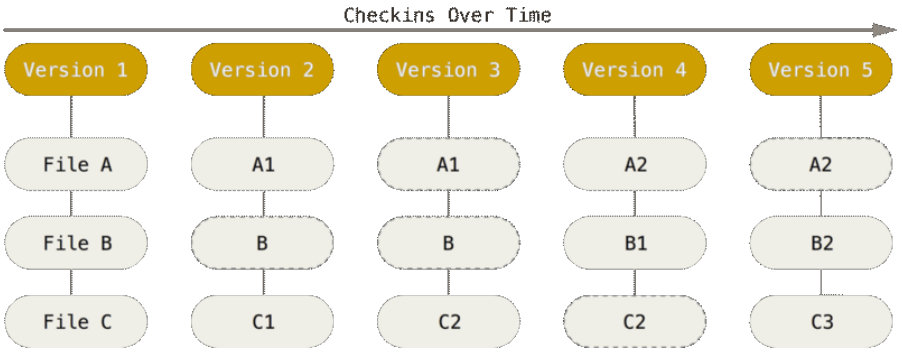
Figure 4.1: *Comparison between centralised and distributed control version systems (VCSs). [Source: Pro Git [45]]*

The main difference between Git and almost any other VCS is the way it considers the data in each version [45]. While other systems use indeed differences between files (deltas) to identify different versions, Git stores each time a *snapshot* of all the files at the moment that the version is created, as shown in Figure 4.2. Of course, there is a 'parent' relationship between files in subsequent versions so that Git does not need to store again files that have not been modified, since it just points at their previous versions [45]. Furthermore, Git allows to perform almost every operation

in local, since the whole project's history is stored locally. A direct advantage of this is that one is able to work on a project without being constantly online [45].



(a) *Storing data as deltas between files*



(b) *Storing data as snapshots of the files*

Figure 4.2: *Comparison between different methods for storing data in version control systems. [Source: Pro Git [45]]*

A new version is created when files are *committed* to the *version database* (which is a Git directory, or repository). This operation is divided into three steps [45]:

- Modification: files are modified in the local *working tree*;
- Staging: modified files are added to a *staging area*;
- Commit: staged files are added to the database and stored.

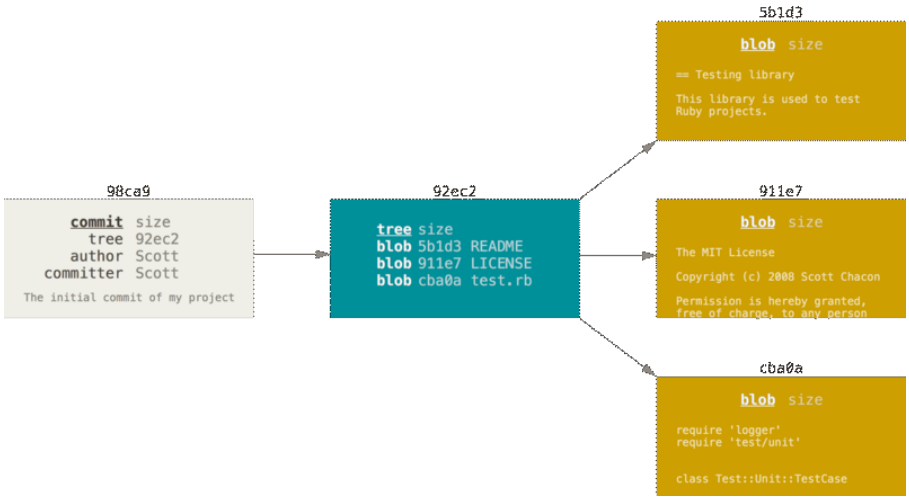
The *working tree* is a *checkout* of one of the *snapshots* (versions) stored in the database, which is pulled out and stored in the working directory on the local disk, while the *staging area* is a file that is part of the Git directory [45]. According to the three steps above, a file that is present in a Git directory can either be *committed*, if stored in a version, *staged*, if has been modified since its last snapshot that had been stored (or if it is a new file) but it is ready to be committed, or *modified* otherwise [45].

4.1.1 Branching and merging

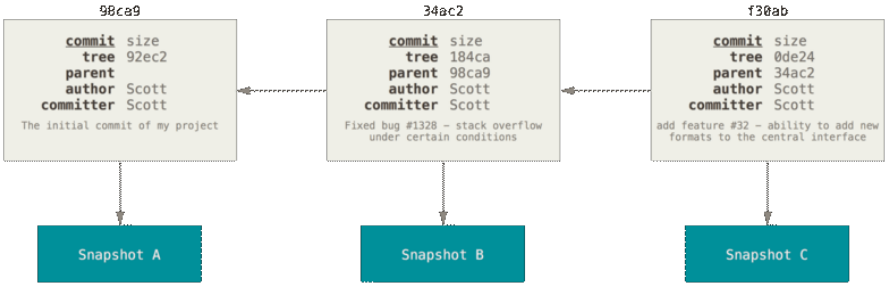
Branching is an important part of any VCS [45], since it allows to create and explore new paths for the development of the project, diverging from the main line, which remains safe and unmodified until the branches are merged. Merging can be considered as the opposite of branching, i.e. the process of reconnecting of one or more branches after having performed and tested some modifications. In order to understand how branching works in Git, it can be helpful to better describe the process of committing. As previously mentioned, Git stores data in a repository as *snapshots* of the files (in other words, their version at a particular moment) that are then staged and committed. The *commit* is actually an object that contains a pointer at: the *blobs*, which are objects that hold the contents of the files and a *tree* that is, in simple terms, a list of the files contained in a directory and of the *blobs* associated to each file's name. Using these elements, the commit is able to recreate the *snapshot* at any time [45]. Moreover, it contains information about the author (name and email), a message which describes the changes introduced with the commit and a pointer at the parent(s), i.e. the commit(s) that are immediately above. The project's development line is in the end a sequence of commits, which are able to reproduce a snapshot of the project at a given time, linked by parent-

relationships. The structure of the commits and of the development line (in a very simple, linear case) are shown in Figure 4.3.

Keeping these considerations in mind, it is possible to easily understand how branches work in Git. A branch is indeed just a *«lightweight movable pointer to one of these commits»* [45].



(a) *Elements of a commit*

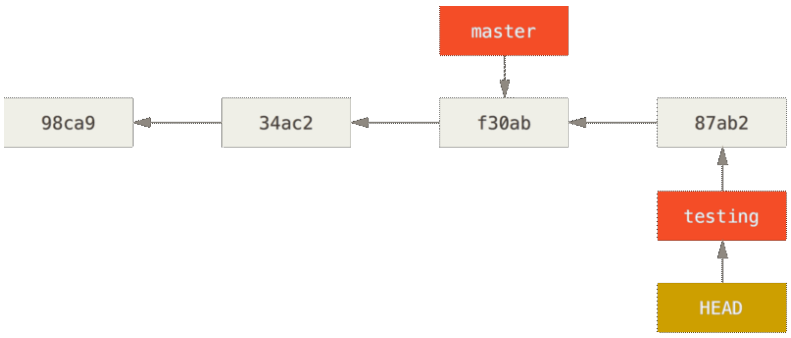


(b) *Sequence of commits*

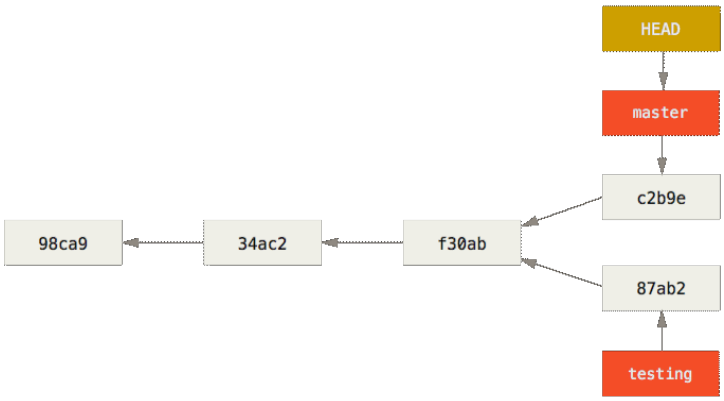
Figure 4.3: *Structure of the commits in Git and of the versions database as a sequence of commits. [Source: Pro Git [45]]*

In Figure 4.4, two quite comprehensive examples of the branching process are provided. In the first one (Figure 4.4a), it can be noticed that the development line is given by a linear sequence of commits. The *master* (or *main*) is the default branch

and it is not pointing at the latest commit in the sequence. Another branch indeed is ahead of the main by one commit. The *head* is instead a pointer at the branch on which one is currently working and it moves when switching from a branch to another one.



(a) *'Linear' branching*



(b) *Divergent history*

Figure 4.4: *Branching process using Git. [Source: ProGit [45]]*

In the second case instead (Figure 4.4b) an example of *divergent history* is provided. This happens when one switches to a new branch, performs some modifications (and commits them) and then goes back to the old branch to do other modifications without having merged them first [45].

Once that a branch has 'served its purpose' (i.e. a given modification has been created and tested) it can be easily merged to another branch, e.g. the main, or just discarded. Git usually manages the merging in a very smooth way. Anyway, some problems may arise when two branches have diverged. If different modifications are performed on the same part of the code in fact, there is a conflict that must be solved before being able to merge the branches.

Thanks to the Git branches' characteristic of being just pointers at the commits, branching is very fast and efficient and it is indeed strongly encouraged by the Git developers [45]. The *main* branch is usually used to store 'final' versions of the project, while a 'development' branch is used to test modifications and new features, creating different branches for each one of them [46]. For further information about Git's functioning and for learning the basic commands for branching and committing, the reader may refer to the ProGit book [45] or the reference guides available at [46].

4.2 Managing the GitHub repository

While Git's functionalities are used to work on the project in local, GitHub is used to share the tool to any potential 'audience', i.e. both users and contributors or, in general and in compliance with the principles of 'free software', whoever is interested in the work for their own purposes. Therefore, after having created a Git repository where the tool (i.e. the collection of files which compose the tool) is stored and developed through a series of commits, it has been uploaded to GitHub. The 'upload' process actually consists in creating a remote repository (hosted, in this case, on GitHub) and linking it to the local one by setting the former as the *remote*, or *origin*, of the latter [45]. The two repositories can communicate easily, through the

'push' and 'pull' commands. In very simple terms, the first one is used to send data from the local repository to its *origin* (the remote repository), the second one vice-versa gets data from the remote and merges them to the local repository[2].

When a remote repository is created in GitHub, it is associated with an URL, which can be used both to share it or to clone its content [47]. The repository hosting the tool is named 'RECOpt'[3] and its content is publicly available to whoever has a GitHub account, which is basically the only requirement to access it. The content of the repository can be accessed in two different ways:

- Cloning, which links a GitHub repository to a local folder;
- Forking, which links two GitHub repositories.

When a remote repository is cloned, its content is downloaded from GitHub to a directory in the local computer [47]. However, the files downloaded are not stored in a local Git repository yet. It is therefore necessary to create one and to stage and commit the files to it. The new local repository is not linked to the original GitHub repository nor it is associated with any *remote*. Even creating a new remote repository in GitHub and linking it to the local repository hosting the cloned files, however, do not restore the link to the original one either. To do this, forking can be used together with cloning. The former is a way to copy an existing GitHub repository keeping a direct connection with it. In fact a forked repository allows to keep its content up to date with the original repository and, above all, allows to collaborate on the project thanks to the 'pull request' mechanism. The latter is a

[2]The process is actually more complex than this and it usually involves a third command, which is 'fetch'. For further information about remote and local repositories, the reader may refer to the ProGit book's chapter 'Working with remotes' [45].

[3]The GitHub repository, owned by cadema-PoliTO, is available at `https://github.com/cadema-PoliTO/RECOpt`.

way to send to the original repository modifications that have been performed on the forked repository. Of course, the owner of the original repository can review the proposed modifications and decided whether to accept (and therefore merge) them or not [47]. After having forked the repository (thus creating a new GitHub repository, linked to the original one) it can be cloned in order to work locally, on a Git repository. The workflow of *fork* and *clone* mechanisms, as well as *pull* and *push*, are summarised in Figure 4.5.

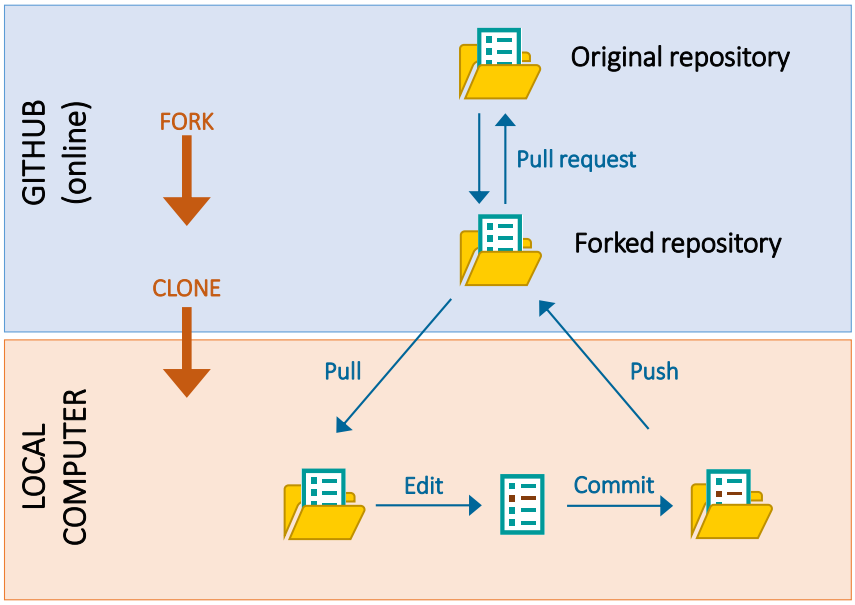


Figure 4.5: *Forking and cloning an existing GitHub repository and usual workflow. [Adapted from Earth Data Science [47]]*

Lastly, it is worth noticing that there is another way of collaborating to a project, i.e. being added as a collaborator. This way, one can directly clone the original (remote) repository into a local one and link the latter to the former. In this case, the 'pull request' mechanism is not necessary since any modifications pushed from the local to the remote (original) repository is directly merged. The reader may find further information about GitHub and the commands to let Git communicate

with GitHub in the Introduction to Earth Data Science textbook [47] or the GitHub documentation, available at [44].

4.2.1 Readme file

A readme file is an important part of any repository in GitHub. It is indeed supposed to be the first file that is seen by the users. The readme (which is usually written in a particular language, Markdown) should contain the name of the repository, and a brief description of what it does. When dealing with code files, requirements for running the code are supposed to be added: in this case, all the codes included in the repository are written in Python 3, which is required for the execution of the files, together with a series of packages. Lastly (and most importantly), the readme file should present the content of the repository, i.e. the names of all the files contained, a brief description of what they do and an indication for the user, whether they are supposed to directly use some files or not. The full readme file of 'RECOpt' is contained in A. The readme, however, is just a 'starter' for the user and is supposed to drive them through the execution of the tool. More comprehensive information is needed for those who want to work on the code rather than just execute it or anyway for those who are interested in the detailed functioning of the tool. This is contained in the tool's documentation that, as previously mentioned, corresponds with 2.

4.2.2 Licence

Licensing is another relevant step when dealing with open-source projects. The licence states indeed the rights and duties of the users with respect to the content of a project. Therefore, a licence is to be included in the GitHub repository, since

standard copyright provisions apply otherwise [48]. When it comes to open-source licences, they should guarantee some basic rights, such as copying, modifying and distributing the content of the repository [48]. Of course one may decide the terms under which licensing a project, but standardised version are to be preferred, because other users are more familiar with them and they clearly states the users' rights and duties [49]. Examples of very popular standardised licences are the GNU General Public Licence (v3.0) and the MIT License. They both allow to copy, modify, distribute the licensed material, also for private or commercial use [49]. Anyway, they differ with respect to the *copyleft*. The latter is, in practical terms, the obligation to redistribute modified (or unmodified) material under the same license [48]. According to the Free Software Foundation, *«copyleft is a general method for making a program or other work free, and requiring all modified and extended versions of the program to be free as well»*[43]. Copyleft licences indeed, such as the General Public Licence, do not allow to make proprietary material out of the licensed material and any modified version cannot be closed-source [43, 48]. On the other hand, the MIT License falls under the 'permissive' licences which have looser terms. The only condition the MIT License poses is indeed to include the copyright and permission notice in any distributed version (modified or unmodified) of the work, even under different licence terms, e.g. without source code [48].

As previously mentioned, the GitHub repository is owned by cadema-PoliTO. Other projects from cadema-PoliTo on GitHub have been distributed using a MIT Licence, therefore, given also the previous considerations, it has been chosen for the repository 'RECOpt' as well. The integral text of the licence and the copyright notice added to the repository are presented in B.

Conclusions

In this thesis, an open-source tool for the evaluation of an energy community or, in general, of a configuration where energy is collectively self-consumed has been presented. In this kind of configurations, self-produced electricity from a renewable installation is virtually shared using the public grid, either instantaneously or through a storage system. A common framework for jointly-acting self-consumers and energy communities has been only recently introduced in Europe. As to Italy, the current provisions have a temporary nature since they are meant to collect evidences before the complete transposition of the European directives. The possibilities for energy communities are hence limited, to a certain extent. Indeed, while in a wider perspective they might participate in a number of energy services, the main focus, at the moment, is on the self-consumption of the locally-produced energy. The latter is quantified in terms of *shared energy*, which is the minimum, in each hour, between the total injections into and the total withdrawals from the grid.

In this context, the tool can be used to provide quick and easy-to-grasp results to simultaneously compare more configurations on the basis of their yearly performances or to evaluate in more detail a single configuration. The two main features of the tool are the simulation of the consumption profiles for an aggregate of household and the optimisation of the power flows realised in the configuration. As to the latter, the production profiles and the simulated consumption profiles are taken as inputs, while the power flows to optimise are the charge/discharge of the battery (storage system) and the feed into/purchase from the public grid. The provisions currently adopted in Italy recognise the economical value of the *shared energy*, which

is incentivised and compensated for the transportation costs. Hence, the optimisation objective can be to maximise the *shared energy*. Alternatively, the interactions with grid can be minimised in order to reduce the excess production that is not self-consumed. The tool has been used to simulate a potential energy community in Northern Italy, composed of 20 households whose yearly energy consumption is around 2400 kWh/year, which is the Italian average. Different sizes of the photovoltaic system and the battery have been simulated, ranging, respectively between 15 and 25 kW_p and between 0 and 40 kWh. However, the aim of the simulation was to discuss the results provided by the tool rather than actually evaluate a case-study.

As to the simulation of the households' consumption profiles, the 'instantaneous' profiles of a random sample of households have been compared with the hourly minimum and maximum and the total consumption profile of the aggregate. As expected, both the transition from 'instantaneous' to hourly values and the aggregation of the households cause a smoothing of the power demand, thus levelling peaks and troughs. Yet, when a small number of households is considered, the aggregate's consumption profile can be quite different from the typical shape assumed for household customers. Even if the appliances' average and total consumption are consistent with the input data, further improvements are required in the simulation of the consumption profiles. For instance, the dependence of some appliances' consumption on the season is to be taken into account in a more accurate way. This is even more true for the air conditioning machinery, especially in the perspective of considering the switching to the electric heating. In addition, while adding other electric appliances to the routine is not a difficult task, generally, accounting for the appliances' consumption in their standby mode might require more elaboration. The latter is the energy absorbed by an appliance when it is not serving its primary

function but is still plugged in and in total it can account for a relevant share of the yearly electricity consumption in a household.

As to the evaluation of the configuration's performances in terms of *shared energy*, first base-case scenarios without the battery have been evaluated. In these cases, increasing the size of the photovoltaic system leads to always smaller increases in the *shared energy*, while an always larger part of the production is fed into the grid. On the other hand, when the battery is included, increasing its size brings an almost linear increase in the *shared energy*, thus in both the self-sufficiency and the self-consumption of the configuration. This is true until the full self-consumption is reached, when the *shared energy* becomes saturated even if the size of the battery is further increased. The simulations performed with the different objectives of maximising the *shared energy* or minimising the interactions with the grid led to the same results. Therefore, under the definition of *shared energy* adopted in the routine, the latter is maximised when the interactions with the grid are minimised. The optimised power flows show that such definition allows to take into account the energy that is shared through the battery only when discharging the excess production stored in it. Lastly, the yearly values of energy that is charged/discharged in the battery show that, depending on its size, the latter might be excessively used thus causing its lifetime to be over before the end of the investment period and requiring to be substituted. This observation raises a relevant issue: when evaluating actual cases, the energetic assessments are to be coupled with an economical analysis of the configurations. Anyway, the results provided by tool are saved in CSV files, to be used for further evaluations.

The tool is open-source and all the codes and the files required for its execution are available on GitHub (at `https://github.com/cadema-PoliTO/RECOpt`) for the users and potential collaborators.

Bibliography

[1] A. Caramizaru and A. Uihlein, "Energy communities: an overview of energy and social innovation," Publications Office of the European Union, Luxemburg, 2020.

[2] European Parliament & Council of the European Union, "Directive (EU) 2018/2001 on the promotion of the use of energy from renewable sources," Dec. 2018.

[3] European Parliament & Council of the European Union, "Directive (EU) 2019/944 on common rules for the internal market for electricity and amending Directive 2012/27/EU," Jun. 2019.

[4] R. J. Hewitt *et al.*, "Social innovation in community energy in Europe: a review of the evidence," *Frontiers in Energy Research*, vol. 7, Apr. 2019.

[5] CEER, "Regulatory Aspects of Self-Consumption and Energy Communities," Bruxelles, Jun. 2019.

[6] T. Bauwens, "Explaining the diversity of motivations behind community renewable energy," *Energy Policy*, vol. 93, pp. 278–290, Jun. 2016.

[7] C. Candelise and G. Ruggieri, "Status and Evolution of the Community Energy Sector in Italy," *Energies*, vol. 13, no. 8, p. 1888, Apr. 2020.

[8] R. Hiteva and B. Sovacool, "Harnessing social innovation for energy justice: A business model perspective," *Energy Policy*, vol. 107, pp. 631–639, 2017.

[9] C. Kunze and S. Becker, "Energy Democracy in Europe: A Survey and Outlook," Rosa Luxemburg Stiftung, Berlin, 2014.

[10] K. Szulecki, "Conceptualizing energy democracy," *Environmental Politics*, vol. 27, no. 1, pp. 21–41, 2018.

[11] B. Kampman, J. Blommerde, and M. Afman, "The Potential of Energy Citizens in the European Union," CE Deltf, 2016.

[12] RESCoop and Friends of the Earth, "Potential for citizen-produced electricity in the EU," 2016.

[13] International Cooperatives Agency, "Statement on the cooperative identity," 4, 1995, pp. 3–4.

[14] Energy & Strategy Group, "Electricity Market Report," Nov. 2020.

[15] J. Heredia, "Clean Energy for All Europeans," (Powerpoint presentation).

[16] ARERA, "Memoria 94/2019/I/COM. [Memoir 94/2019/I/COM],"

[17] IRENA (2020), "Innovation landscape brief: Peer-to-peer electricity trading," International Renewable Energy Agency, Abu Dhabi.

[18] GSE, "Rapporto Statistico - Solare Fotovoltaico 2018. [Statistical Report - Solar Photovoltaic 2018]," Jun. 2019.

[19] A. Cielo, *Le comunità energetiche: simulazione ed ottimizzazione tecnico-economica.* [Energy communities: techno-economic simulation and optimisation], 2020.

[20] Governo della Repubblica Italiana, "Testo coordinato del decreto-legge 30 dicembre 2019, n. 162. [Coordinated text of the Decree-Law of 30 December 2019, n. 162]."

[21] ARERA, "Delibera 318/2020/R/EEL. [Resolution 318/2020/R/EEL],"

[22] GME. (2021). "Gestore dei mercati elettrici. [electricity markets operator]," [Online]. Available: `https://www.mercatoelettrico.org/it/`.

[23] ARERA. (2021). "Prezzi e tariffe. [prices and tariffs]," [Online]. Available: `https://www.arera.it/it/prezzi.htm`.

[24] Ministero dello Sviluppo Economico, "Decreto 16 Settembre 2020. [Decree of 16 September 2020]," Nov. 2020.

[25] RSE. (2020). "DOSSIER - Gli schemi di Autoconsumo Collettivo e le Comunità dell'Energia [Collective Self-Consumption schemes and Energy Communities]," [Online]. Available: `https://dossierse.it/17-2020-gli-schemi-di-autoconsumo-collettivo-e-le-comunita-dellenergia/`. (last access on 02.02.2020).

[26] GSE, "Gruppi di autoconsumatori di energia rinnovabile che agiscono collettivamente e comunità di energia rinnovabile - Regole tecniche per l'accesso al servizio di valorizzazione e incentivazione dell'energia elettrica condivisa. [Technical rules for the access to the valorisation and incentive of the shared energy]," Dec. 2020.

[27] GSE, "Rapporto Statistico - Solare Fotovoltaico 2019. [Statistical Report - Solar Photovoltaic 2019," Jun. 2020.

[28] JRC. (2019). "Photovoltaic Geographical Information Tool," [Online]. Available: `https://re.jrc.ec.europa.eu/pvg_tools/en/`. (last access on 03.02.2020).

[29] IRENA, "Electricity Storage and Renewables: Costs and Markets to 2030, International Renewable Energy Agency," 2017.

[30] Wikipedia. (2021). "Levenshtein distance," [Online]. Available: `https://en.wikipedia.org/wiki/Levenshtein_distance`. (last access on 11.03.2020).

[31] Interreg Med - StoRES. (2020). "PV and Storage Optimization Tool," [Online]. Available: `http://www.storestool.eu/`. (last access on 10.02.2020).

[32] F. Di Andrea and A. Danese, "MICENE - MIsure dei Consumi di ENergia Elettrica in 110 abitazioni Italiane: Curve di carico dei principali elettrodomestici e degli apparecchi di illuminazione. [Measuring the Electricity Consumption in 110 households in Italy: Load profiles of the main electric appliances and lighting devices]," eERG, Sep. 2004.

[33] Enea. (2016). "KiloWattene: analisi dei consumi elettrici residenziali italiani e distribuzione statistica dei valori di consumo annuo [Analysis of the electric consumption in Italian households and statistical distribution of yearly consumption values]," [Online]. Available: `http://kilowattene.enea.it/KiloWattene-consumi-famiglie.html`. (last access on 14.01.2020).

[34] eErg. (2010). "MICENE - MIsure dei Consumi di ENergia Elettrica nel settore domestico [Measuring the Electricity Consumption in households]," [Online]. Available: `https://www.eerg.it/index.php?p=Progetti_-_MICENE`. (last access on 14.01.2020).

[35] ISR - University of Coimbra, "Residential Monitoring to Decrease Energy Use and Carbon Emissions in Europe," Intelligent Energy Europe Programme, Sep. 2008.

[36] F. Ferro, *Studio analitico-sperimentale di un sistema fotovoltaico con accumulo e gestione smart dei carichi elettrici.* [Analytical and experimental study of a photovoltaic system combined with a storage system and smart managing of electric loads], 2016.

[37] CESI, "Contributo delle elettrotecnologie per usi finali al carico di punta. [Contribution to the peak load by end-use electric appliances]," Jun. 2005.

[38] CESI, "Individuazione delle tendenze della domanda e delle tecnologie per usi finali legate ad esigenze di benessere. [Identifying trends in the demand and in end-use technologies connected to welfare needs]," Dec. 2004.

[39] ISTAT, "Indagine sui consumi energetici delle famiglie: principali risultati. [Investigation on the energetic consumption in households: main results]," Dec. 2014.

[40] Enea. (2020). "Dipartimento unità per l'efficienza energetica," [Online]. Available: `https://www.efficienzaenergetica.enea.it/servizi-per/cittadini/interventi-di-efficienza-e-risparmio-energetico-nelle-abitazioni/etichetta-energetica/etichetta-energetica-apparecchi.html`. (last access on 14.01.2020).

[41] S. Sibilio, A. D'Agostino, F. M., and C. M., "Ricerca di Sistema Elettrico - Valutazione dei consumi nell'edilizia esistente e benchmark mediante codici semplificati: analisi di edifici residenziali. [Assessments about the consumption in existing buildings and benchmarks through simplified codes: analysis of residential buildings]," Enea, Mar. 2009.

[42] E. K. P. Chong and S. H. Zak, "An Introduction to Optimization, Second Edition," in. 2001, ch. Introduction to Linear Programming.

[43] Free Software Foundation. (2018). "GNU Operating system - Philosophy of the GNU project," [Online]. Available: https://www.gnu.org/philosophy/philosophy.html. (last access on 02.03.2020).

[44] GitHub, Inc. (2021), [Online]. Available: https://github.com/. (last access on 02.03.2020).

[45] S. Chacon and B. Straub, *Pro Git, Second Edition*. Apress, 2014. [Online]. Available: https://git-scm.com/book/en/v2.

[46] S. Chacon. (). "git," [Online]. Available: https://git-scm.com/. (last access on 02.03.2020).

[47] L. Wasser and J. Palomino, "Introduction to Earth Data Science," in, ch. Git and GitHub. [Online]. Available: https://www.earthdatascience.org/courses/intro-to-earth-data-science/git-github/.

[48] GitHub. (2021). "Licensing a repository," [Online]. Available: https://docs.github.com/en/github/creating-cloning-and-archiving-repositories/licensing-a-repository. (last access on 02.03.2020).

[49] Open Source Guides. (). "The Legal Side of Open Source," [Online]. Available: https://opensource.guide/legal/#which-open-source-license-is-appropriate-for-my-project. (last access on 02.03.2020).

Appendix A.

RECOpt - Readme

Energetic evaluation and optimisation of renewable energy communities

The repository contains a tool that optimises the operation of a PV system with energy storage for fixed or variable (parametric) sizes for both of them, in the context of collective self-consumption and energy communities in Italy. PV production data are to be provided by the user (PVGIS database can be used), while consumption profiles are generated for an aggregate of households using probabilistic methods.

Requirements

Codes included in this repository are written in Python 3, that is the only real requirement. They have been tested with Python 3.8 but also earlier version of Python 3 should work. Python packages needed for running the methods are: pathlib, numpy, scipy, pulp, csv, tabulate, matplotlib.pyplot, math, random. All the other self-created methods needed for the tool to work are contained in this repository.

Content of the repository

Input/

In this folder, all the input *.csv* files needed for the calculation are contained. Some of them must be updated from the user. Their name is properly formatted so that each methods knows which file to look at when certain data are needed. Particularly, the files are the followings.

Optimisation and energy assessment of the PV-storage system

These files should be updated by the user.

- *'pv_production_unit.csv'*: it contains the hourly unit production from the PV installation(s) in the given location, for typical days (one for each month of the year)

- *'battery_specs.csv'*: it contains the specifications for the battery (SOCmin, SOCmax, efficiencies,...)
- *'PVGIS_data.csv'*: it contains the hourly production from a PV installation in a given location for a number of years, as downloaded from PVGIS. It can be used to obtain *'pv*production*unit.csv'* if the latter is not provided from the user.

Generation of the load profiles

These files don't need to be updated by the user.

- *'eltdome_report.csv'*: it contains the attributes for all the appliances. It also contains, for each appliance, its "nickname", that is crucial for the correct loading of the other files.
- *'classenerg_report.csv'*: it contains the yearly energy consumption for each appliance, according to the different energy classes.
- *'coeff_matrix.csv'*: it contains the "user's behavior" coefficient for each appliance, according to the different seasons.
- Average daily load profile files for a type of appliance. They contain the time (in hours, from 00:00 to 23:59), generally with a resolution of 10 min, in the first column and the average power demand in Watt in the second column. Their name is formatted as follows: `'avg_loadprof' + '_' + app_nickname + '_' + seasons + '_' + day + '.csv'`. Where seasons indicates if the load profile is different according to the season (*'w'* stands for winter, *'s'* for summer, *'ap'* for autumn/spring) or the same for each season (*'sawp'*) and day indicates if the load profile is different according to the day in the week (*'wd'* stands for weekday and *'we'* for weekend day) or the same throughout the whole week (*'wde'*).
- Duty cycle files. They contain the time, with a resolution of 1 min, in the first columns and the power demand in Watt from the appliance in the second column. Their name is formatted as follows: `'dutycycle' + '_' + app_nickname + '.csv'`

Python files

- `main.py`: this is the only file that should be directly used by the user. No manual modifications should be made, e.g. to change some parameters; the user just needs to make it run. Parameters are updated from keyboard and stored in Parameters/. Results, both *.csv* files and *.png* figures, are stored in Outputs/ for each in simulation, respectively, in Files/ and Figures/.

- `pvgis_to_csv.py`: the module processes the data about hourly production from the PV, contained in a *.csv* file downloaded from PVGIS. This module should be used from the user if the *'pv*production*unit.csv'* file is not directly provided.

- `shared_energy_evaluator.py`: the module contains a method that evaluates the performance (shared energy, and other quantities) for a given configuration (number of households, size of the PV and battery systems) in one year, using a number of typical days (two for each month, both week-day and weekend-day). This module is not directly used by the user.

- `battery_optimisation.py`: the module contains a method that optimises the operation of the battery in one day, once that the production from the pv and the consumption from the households are given. At the moment, if the user wants to change the objective of the optimisation, this should be done here, manually. This module is not directly used by the user.

- `aggregate_load_profiler.py`: the module contains a method that generates the aggregated load profiles in different typical days (two for each season, both week-day and weekend-day) for a number of households. If the user chooses so when running the simulation, detailed files and figures about the load profiles and the energy consumption from the electric appliances are generated and saved in Output/. If the user wants to change some very specific parameters about the generation of the load profiles, this should be done here. Normally the user does not need to use this module, but it can be used to test the generation of the load profile for the aggregate of household.

- `house_load_profiler.py`: this module contains a method that computes the load profile for a household in a given typical day. Normally the user does not need to use this module, but it can be used to test the generation of the load profile for a single household.

- `load_profiler.py`: this module contains a method that computes the load profile for a single appliance in a given typical day. Normally the user does not need to use this module, but it can be used to test the generation of the load profile for a single appliance.

- `load_profile_aggregator_trapz.py`: this module contains a method that aggregates some profiles using a different time-step. This module is not directly used by the user.

- `plot_generator.py`: this module contains all the methods for the creation of the figures showing the results. This module is not directly used by the user.

- `parameters_input.py`: this module contains the methods that are used in main for updating the paramters value to the user's keyboard input. The parameters are saved in Parameters/. This module is not directly used by the user.
- `levenshtein_distance.py`: this module contains a method that uses Levenshtein distance between two string to suggest the closest match to a word (user's input) and a list of words. This module is not directly used by the user.
- `datareader.py`: this module contains different methods that properly read the various input files. This module is not directly used by the user.
- `tictoc.py`: this module contains two methods that are a Python adaptation of MatLab's tic-toc functions. This module is not directly used by the user.

Appendix B.

RECOpt - MIT Licence

Copyright (c) [2021] [cadema-PoliTO]