POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Controlling Lithium-Ion Batteries Through Reinforcement Learning

Tesi di Laurea Magistrale in
Computer Science and Engineering - Ingegneria Informatica

Oscar Francesco Pindaro, 946671

Advisor:
Prof. Marcello Restelli

Co-advisors:
Marco Mussi
Francesco Trovò

Academic year:
2020-2021

Abstract: Smart Grids are the evolution of the traditional electric grid and allow a two-way flow of electricity and information between different actors. At the edge of this network, consumers can produce energy with photovoltaic panels and satisfy their energy consumption needs autonomously. Due to the intermittent nature of solar energy production, these units are characterized by periods of energy surplus and others of energy deficit. To solve this problem, Lithium-Ion battery packs are used to store energy in excess for later use and reduce expensive energy requests to the electric network. However, these accumulation systems are characterized by a degradation process that reduces their capacity and performance. In this work, we develop a Reinforcement Learning controller optimizing energy management policy by balancing the use of the battery packs and the energy network to reduce economic losses. More specifically, we design a system to learn a storage/consumption strategy able to balance between the degradation of the battery and the economic value of trading energy. The work resulted in a policy allowing to reduce the economic loss of 15% w.r.t. state-of-the-art controllers.

Key-words: Reinforcement Learning, Smart-Grids, Lithium-Ion Batteries, Fitted Q-Iteration, Control, State Of Health

1. Introduction

A *Smart Grid* [1] is an electric grid integrated with Information Technologies that allows to monitor, manage and repair the electric network, resulting in more efficient energy distribution and overall greater reliability and availability of the electric systems. Historically, the electric grid has had a simple structure, and it can be divided into four parts:

- *Energy Producers*: a low number of energy producers generate electric power with chemical or nuclear plants and make it available in the transmission grid. The plants are characterized by monumental dimensions due to the high costs linked to the gathering of raw materials and the exploitation of economies of scale. A single plant is responsible for a vast number of users.
- *Trasmission Grid*: efficiently transfers energy over long distances. This is done by converting and stepping the voltage up to high values.
- *Distribution Grid*: distributes and delivers electricity to the final users. It should keep up with the energy demand and monitor energy uses.
- *End Users*: consume passively electric energy.

A Smart Grid enhances the functionalities of a traditional grid by leveraging information technologies. The main change of paradigm is given by the support of a *two-way flow* of electricity and information. Indeed, it can detect and react to events occurring in the grid, such as power generation, transmission, distribution, and consumption. In order to achieve this, the Smart Grid is composed of loosely coupled control subsystems that exchange information and interact with each other.

In this setting, end users can produce and sell energy for monetary compensation and even form independent communities called *micro-grids* that can disconnect momentarily from the grid and sustain their consumption autonomously. In the case of a failure at the edge of the network, end-users and energy communities can help the Smart Grid to solve the energy problem by intervening promptly and removing the burden from the energy producers. In this way, the grid infrastructure can be designed for lower peaks and is, therefore, less expensive. In this context, solar energy production becomes a valid and cheap alternative to traditional sources such as fossil fuels [2]. The lower entrance barriers w.r.t. the thermal and petrol-chemical sectors open the possibility of producing energy at different scales, from domestic to industrial use cases. This kind of energy production has the advantage of being a source of inexhaustible free energy, but its availability is not controlled by market conditions or third party actors. However, solar energy comes with its own limitations. Its availability varies highly due to weather changes, and, therefore, it is challenging to predict its future availability, even relying on weather forecasting services.

Notice that, in the Smart Grid, end users can install *photovoltaic panels* for a variety of reasons. For example, one could reduce energy consumption from the grid or generate revenue by exploiting fluctuations in energy prices. Energy production and peak user demand are not aligned, and therefore, accumulation systems are used to store energy surpluses to meet future demand. The use of accumulation systems greatly enhances the possibility of lowering energy costs and reaching more stable energy independence.

In a domestic environment, controllers are designed to decide how to store in the accumulation system the energy generated by the PV. These controllers generate a profit by meeting the domestic system energy demand with the previously generated energy, and by selling energy in excess to the Smart Grid. These behaviours need to take into account three main challenges. The first one is *Energy Arbitration*. An arbitration is the purchase and sale on a particular asset aimed at generating a profit from variations in the listed price of the asset. In order to perform arbitration, the controller needs to make predictions on future market prices, and understand which are the most profitable moments for selling energy. *Weather Forecasting* is fundamental when dealing with solar energy production. Indeed, a controller should be able to predict energy availability: this allows the house to never depend on expensive energy purchased from the electric grid. The last challenge is the *Degradation* that accumulation systems are subject to. Indeed, they are mainly composed of Lithium-Ions battery packs, a very efficient and high-energy-density battery technology, and are affected by a degradation process that lowers their capacity and efficiency over time, caused by the natural aging that each battery incurs, environmental impacts (such as storing conditions), and the dynamic loading. Battery degradation is a highly non-linear process, which progresses at different rates in different moments of the life of the battery.

Therefore, the profit of energy production depends on conflicting factors: a controller should be able to store energy for future uses, while avoiding too intensive battery cycling. A battery purchase is very burdensome, and it is crucial to find the best way to control the battery and generate as much profit as possible.

Original Contribution The novelty of this work is the design of a controller to solve simultaneously the three challenges above mentioned, with a focus on battery degradation management. Long-term profit maximization is achieved by taking into account the revenue generated with energy arbitrage and the cost caused by battery degradation. The proposed method is able to amortize the battery cost on an unknown time horizon, since battery life is heavily influenced by cycling conditions. The controller also performs weather

forecasting by taking into account the daily and annual periodicity. Finally an interpretation of the behaviour of the controller is also discussed, allowing to understand better which are the relevant physical quantities that contribute in the generation of a higher profit. These considerations allow the controller to generate up to 15% more in profit with respect to state of the art techniques.

Thesis Structure The thesis is organized in the following way. Section 2 contains the theoretical background required to understand the remainder of the thesis and discuss the existing literature on the topic. The problem formulation and the solution are presented in Section 3 and 4 respectively. Finally, Section 6 concludes the thesis.

2. Background and Related Works

In this section we present the theoretical background on the lithium-ion batteries, focusing specifically on models for the degradation of the battery, and an algorithm to evaluate the battery consumption. Finally, we provide the basis of Reinforcement Learning, which will be employed in the following sections to learn optimal battery management policies.

2.1. Lithium-Ion Batteries

Chemistry Lithium-Ion batteries are used to convert electricity into chemical energy and vice versa. This technology allows great flexibility, since energy surplus can be stored in the battery and it allows to create compact and portable electronic devices. A battery cell is a stacking of three main components: the anode, the electrolyte, and the cathode [3]. The *anode* is usually composed of a metallic element that can be easily oxidized and can produce ions and electrons. Once electrons are freed with a chemical reaction, they are conducted through the metal body and the anode becomes negatively charged. The metallic element is infused with Lithium ions. Lithium has the most negative reduction potential and it is also able to generate great electro-chemical capacities, due to its low atomic weight. This two factors allow to design light-weight, energy dense batteries with very high anode oxidation. A battery *electrolyte* is used to separate the anode from the cathode, so that they don't interact directly. However, it is made of a porous material that allows Lithium ions to pass through it. The *cathode* is where a reduction reaction consumes electrons and positive ions, releasing energy. It is made of a material that allows high energy reactions and consequently high voltages.
During a battery discharge, electrons leave the battery from the anode, and are discharged on an external loaded circuit. The Lithium ions then migrate through the electrolyte to the cathode, generating a current inside the battery. These reactions are generally catalyzed with the help of a solvent that is able to dissolve the Lithium salts and produce a solution with a high ionic conductivity.

Characterization The main physical quantities that characterize a battery are State of Charge (SoC), Voltage (V), Current (I) and State of Health (SoH), formally defined as follows:

Definition 1 (SoC)**.** *State of Charge (SoC)* $\sigma_t \in [0, 1]$ is the amount of battery capacity currently available defined as [4]:

$$\sigma_t = \frac{C_t}{C_{t,max}}, \tag{1}$$

where C_t is the remaining charge in the battery, and $C_{t,max}$ is the fully charged battery capacity at time t. $C_{t,max}$ depends from time since batteries are subject to a degradation process that lowers its maximum capacity over time. SoC is a pure number.

The SoC evolves through time following the equations:

$$\sigma_t = \sigma_0 - \frac{1}{C_{t,max}} \int_0^t I_t d\tau, \tag{2a}$$

$$\sigma_{t+1} = \sigma_t - \frac{I_t \Delta t}{C_{t,max}}, \tag{2b}$$

where I_t is the current applied on the battery at time step t. Notice that Equation (2a) is the integral form and needs the starting condition σ_0, while Equation (2b) expresses the SoC with a discrete differential time form.

Definition 2 (Voltage)**.** *Voltage (V)* is the difference in electric potential between two points, which is defined as the work needed per unit of charge to move a test charge between the two points. The unit of measurement is Volt.

Voltage is a fundamental characteristic of a battery and is determined by the chemical characteristics of the battery. A battery is rated at a given nominal voltage V_{max}. Multiple models can be used to have an approximation of the voltage exposed by the battery. V_{OCV}, the OpenCircuit Voltage, is the difference of electric potential between the two terminals of the battery when no load is present, and therefore the battery is disconnected from any circuit. V_{OCV} has a polynomial and exponential dependency with SoC:

$$V_{OCV} = a_0 + a_1\sigma + a_2 * \sigma^2 + a_3\sigma^3 + b_0 e^{b_1 \sigma}. \tag{3}$$

Definition 3 (Current)**.** *Current (I)* is defined as the number of charges passing through a point in a unit of time. The unit of measurement is Ampere.

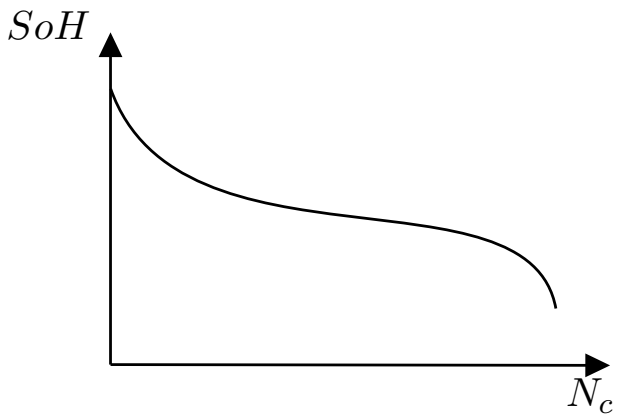


Figure 1: Relationship between cycle number and SoH [12].

Current directly influences SoC values, as already showed in Eq. (2), and is linked to voltage with the following relationship:

$$P_b = VI, \tag{4}$$

where P_b is the power generated or absorbed by the battery.

Definition 4 (SoH)**.** *State of Health (SoH)* measures the remaining capacity of a battery and is defined as:

$$SoH_t = \frac{C_{t,max}}{C_{0,max}}, \tag{5}$$

where $C_{t,max}$ is the capacity at time t and $C_{0,max}$ is the rated capacity of the battery. SoH is a pure number.

Batteries are subject to a degradation process that lowers their overall capacity overtime, and the real capacity quickly moves away from the nominal value. SoH evolution is a highly non-linear process that is caused by a variety of factors. Most of the degradation is concentrated at the beginning and end of the battery life, with a heavy slow down in the battery degradation rate during its mid-life. Degradation is caused by irreversible reactions between the anode and the electrolyte. Capacity fade is a consequence of the irreversible consumption of lithium ions that cause the creation of the Solid Electrolyte Interphase (SEI), a layer of non-reactive compounds that limits the amount of Lithium ions that can be exchanged between anode and cathode. Degradation D_t can be also used to describe the battery health:

$$D_t = 1 - SoH_t. \tag{6}$$

Modelization Parameters estimation such as SoC and SoH are still an open problem [5]. Multiple works have been proposed, and they can be divided into 3 main groups:

- *Physics Based electro-chemical models*: the characteristics of the battery are expressed with the use of static and dynamic equations. The main drawback of this techniques is that they are not suitable for real-time application due to their large number of unknown variables. Moreover, they run into over-fitting or local optimization problems. They also need a very detailed model, otherwise the simulation results may not be truthful. Examples are the pseudo-two-dimensional models (P2D) such the work in [6], that laid the foundations for this type of modelling, and the single-particle model (SP) [7], that is more rigorous and accurate but has no analytical solutions.
- *Electrical equivalent circuit models*: Battery behaviour is described through electrical components. The main advantage of this approach is that it simplifies the structure of the battery and allow fast and efficient computations , allowing real-time simulation. The simplest circuit model is the Rint model [8], which is composed of an ideal generator in series with a resistor, while a more complex modelling can be achieved by considering a first-order resistor-capacitor (RC) model [9, 10].
- *Data-driven models*: This models [11] use a regressor trained on real-life or synthetic data to estimate the parameters. However, the dataset can at the same time give great flexibility but easily influence the results due to over-fitting.

Degradation Model A degradation model allows to simulate the dynamics that causes a battery to loose its capacity over time. This work uses the model proposed by Xu et al. [13], a data driven model that combines the theoretical considerations about Lithium-Ion battery chemistry with experimental observations. It has the

main advantage of being applicable to different operating conditions. Battery degradation is a non-linear process that depends both on time and stress cycles. Indeed, it is linked to factors such as charging, discharging, time and temperature, but also its current state of life.
These qualitative considerations can be formalized in two stress functions: calendar and cycling ageing. More specifically, *calendar ageing* is the degradation stress that a battery suffers independently from its use. It depends on the operational life of the battery, the mean SoC and the mean temperature at which it is preserved. Instead, *cycling ageing* is caused by the direct use of the battery. Every cycle is modeled as a single stress event **independent** from the others, and the accumulated degradation is the sum of the capacity reduction caused by each cycle. The overall stress f_d is a linear combination of calendar and cycling ageing. Formally:

$$f_{cal} = f_t(t, \bar{\sigma}, \bar{T}), \tag{7a}$$

$$f_{cyc} = \sum_{i}^{N} n_i f_c(\delta_i, \sigma_i, T_i), \tag{7b}$$

$$f_d = f_{cal} + f_{cyc}, \tag{7c}$$

where $\bar{\sigma}$ and $\bar{T}$ are respectively the mean SoC and temperature at which the battery as been stored, t is the age of the battery. In Equation (7b), N is the number of equivalent cycles, δ_i , σ_i and T_i are the Depth of Discharge, mean State of Charge and Temperature of the i-th cycle, and n_i indicates whether cycle i is a full of half cycle. These values are computed through the a cycle counting algorithm that converts irregular SoC profiles into a sequence of standardized cycles.
Battery degradation can be then expressed as:

$$D = 1 - \alpha_{sei} e^{-f_{sei}} - (1 - \alpha_{sei}) e^{f_d}, \tag{8a}$$

$$f_{sei} = \beta f_d. \tag{8b}$$

Equation (8) suggests that the degradation is non-linear with respect to the overall stress factor f_d. Indeed, a battery suffers from high degradation rates at the beginning of its life, then it meets a plateau and then the degradation increases rapidly when it reaches its end of life. It also considers the fast degradation cause by the SEI, whose formation rate decreases when a stable film has been formed. Therefore, the equation can be divided into two component: one that takes into account the capacity loss caused by the SEI formation, and one that considers capacity fading at a rate proportional to the battery life. Equation (8b) indicates that the SEI formation is proportional to the battery used. These formulas only apply to fresh batteries.

2.2. Rainflow Cycle Counting

A cycle counting algorithm is used to extract from a loading history a series of cycles. This algorithm is used in physical systems subject to hysteresis and degradation and it associates to each extracted cycle a closed stress-strain hysteresis loop. The rainflow algorithm [15, 16] name is derived from a analogy between how this method works and the rain falling on a pagoda roof. The cycles are computed in the following way:

1. The loading profile is rotated 90 degrees such that the time axis is vertical pointing downward.
2. From each inner extremity, a flow of rain is generated.
3. The rains keeps pouring down the profile until:
   (a) it falls on a previous wider cycle;
   (b) it meets a previous flow falling from above;
   (c) it falls down the roof.
4. Each cycle is identified by pairing up the fall with same characteristics.

In this work, the rainflow algorithm is applied to the SoC profile of a Lithium-Ion battery. The output are N cycles, each characterized by its mean SoC, its Depth of Discharge, the starting time and the cycle duration. An example of the output provided by the rainflow algorithm is presented in Figure 2. The plot has been rotated, with the y axis oriented left to right. On this axis the variable associated to the stress variable is represented. If only ascending cycles are considered, cycle $0-1$ stops at the second roof, since point 2 has a lower value of point 1. The same holds for cycles 2-3 and 4-5. Cycle 8-9 is interrupted by the falling stream of rain since the previous cycle 6-7 start before point 8. Cycle 6-7 is the deepest and therefor is never truncated by another flow. The same consideration can be applied to descending cycles.

2.3. Reinforcement Learning

Machine Learning (ML) [17] is the discipline that studies the algorithms and techniques that allow a program to learn from experience and become proficient in a specific task using that experience. This field can be divided in

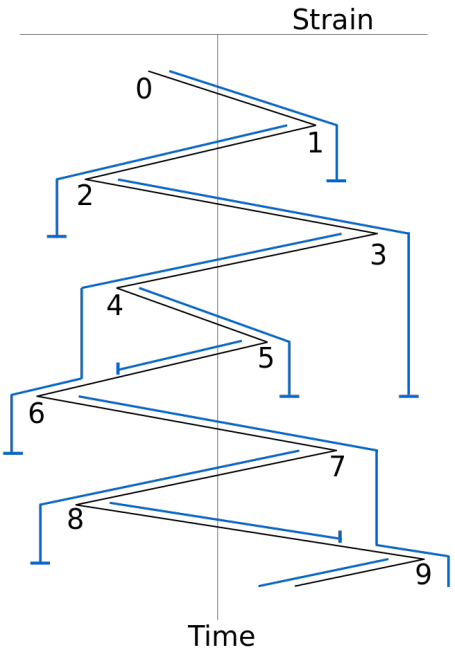


Figure 2: Visualization [14] of the rainflow cycle counting algorithm.

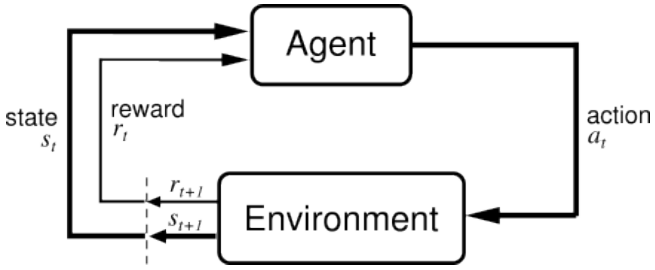


Figure 3: The agent-environment interaction in reinforcement learning.

three main groups called Supervised Learning, Unsupervised Learning and Reinforcement Learning (RL). While Supervised learning is based on inductive inference and operates with labelled data, and Unsupervised Learning analyzes and clusters unlabeled data, Reinforcement Learning (RL) [18] is the discipline that studies sequential decision making. A RL agent is trained through interaction with the world, so that it finds the best sequence of actions that maximizes the cumulative value of a signal called reward. In what follows, the background needed to formalize the problem of controlling an unknown energy management system is formalized.

Agent-Environment Interface The Agent-Environment Interface is an abstraction used to formulate RL problems. The decision maker is called *agent*. Everything external to the agent is called *environment*, and the agent interacts with it. Agent and environment communicate in a *closed loop*: every decision step, the agent measures the environment and performs an action based on that measure. After an action, the environment emits a reward and the system evolves to the next state. The agent has the goal of maximizing the reward coming from the environment by creating a mapping between states and actions. This framework allows to describe multiple settings thanks to its high level view: the state can be composed by a combination of physical variable or categorical concept, and actions can coordinate an actuator or they can be a high level concept. States can also be discrete or continuous, and can encode characteristics of the world and model memory of past events. The mathematical model that best suits this abstraction is the Markov Decision Process (MDP).

Markov Decision Process Formally, an MDP $\mathcal{M}$ is defined as a tuple:

$$\mathcal{M} := \langle \mathcal{S}, \mathcal{A}, \mathcal{P}(s'|s,a), R(s,a), \gamma, \mu_0 \rangle, \tag{9}$$

where:

- $\mathcal{S} \subseteq \mathbb{R}^n$: set of states, which contains all the possible states characterizing the system under analysis.
- $\mathcal{A}$: set of actions that the agent can perform. The set of possible actions may depend on the current state.
- $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$: state transition function, specifing the probability to go to state s' for each generic state/action (s, a) pair. Depending on the characteristics of this function, it can represent deterministic or stochastic environments.
- $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$: reward function, defining for each state/action pair (s, a) the expected immediate reward obtained.
- $\gamma \in [0, 1]$: the discount factor, which describes how much the controller weights future rewards.
- $\mu_0 \in [0, 1]^{|\mathcal{S}|}$: Distribution of initial states. It describes for each state s the probability to start an episode in that state.

This abstraction is used to model the interaction of the agent with the environment. This interaction may have a fixed or maximum length, and, therefore, the agent is performing an *episodic task*. On the contrary, if the interaction goes on for an undefined amount of time, the task has an *indefinite time horizon*. In this last case, the cumulative reward is not bounded and will diverge for environments without absorbing states that stop the agent-environment interaction. The discount factor γ is used to avoid this problem and also to model the importance of future reward for the agent. A low γ will result in an *myopic* agent that will learn to maximize immediate rewards and, while a high value of γ will emphasize the weight of future reward, and the agent will be *farsighted*.[1]
Given this framework, the goal of the agent is to **maximize the cumulative discounted reward** G_t:

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}. \tag{10}$$

A very important characteristic of an MDP is the *Markov property*, expressed as follows:

$$Pr(s_{t+1}|s_{0:t}, a_{0:t}) = Pr(s_{t+1}|s_t, a_t). \tag{11}$$

The Markov property states that the probability of falling in the next state does not depend on the history of the states visited by the agent and the actions performed, but is only affected by the last state and the last action.
All reinforcement learning algorithms try to estimate *value functions*, which quantify how good is for an agent to be in a certain state s and perform a certain action a. The values of the value function are strictly linked to how the agent is choosing its action. This behaviour is called *policy*. Formally, a *policy* is a mapping $\pi(a|s) : \mathcal{S} \times \mathcal{A} \to [0, 1]$ from states to probability of playing every action. The *state-value function* of a state s under the policy π is defined as:

$$v_\pi(s) = \mathbb{E}\Big[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|s_t = s\Big]. \tag{12}$$

Similarly, the *action-value function*, defined as the cumulative discounted reward gained after playing action a in state s and then keeping following policy π, is defined as:

$$q_\pi(s, a) = \mathbb{E}\Big[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|s_t = s, a_t = a\Big]. \tag{13}$$

The objective of an algorithm is to encode this functions in some fashion, possibly using a much lower number of parameters with respect to the number of states, and find the best policy on this encoding.
A fundamental property of these value functions is that they satisfy a recursive relationship called *Bellman equation*:

$$v_\pi(s) = \mathbb{E}_\pi[G_t|s_t = s] = \sum_a \pi(a|s) \sum_{s'} P(s'|s, a)\big[r_{t+1} + \gamma v_\pi(s')\big]. \tag{14}$$

This equation links the value function of a state to the values of its successor states. It states the the value of a state is equal to the reward obtained plus the discounted value of the subsequent states. In order to solve the MDP, we need to find the *optimal policy* such that the value function is maximized for every $s \in \mathcal{S}$. The *optimal state-value function* v^* is defined as:

$$v^*(s) = \max_\pi v_\pi(s) \qquad \forall s \in \mathcal{S}. \tag{15}$$

[1] From now on, the value of a generic variable x at time t will be denoted as x_t. The history values from time step $t1$ to $t2$ will be denoted as $x_{t1:t2}$

Optimal policies also achieve *optimal action-value function* q^*:

$$q^*(s,a) = \max_\pi q_\pi(s,a) \qquad \forall s \in \mathcal{S}, a \in \mathcal{A}. \tag{16}$$

This last equation (16) can also be written incorporating v^*:

$$q^*(s,a) = \mathbb{E}\Big[r_{t+1} + \gamma v^*(s_{t+1}|s_t = s, a_t = a)\Big] \qquad \forall s \in \mathcal{S}, a \in \mathcal{A}. \tag{17}$$

Since v^* and q^* are value functions for a policy, they must satisfy the consistency property expressed in (14), and therefore we can write for both of them the *Bellman optimality equations*:

$$v^* = \max_a \mathbb{E}_{\pi^*}\Big[\sum_{s'} P(s'|s,a)\big[r + \gamma v^*(s')\big], \tag{18}$$

$$q^* = \sum_{s'} P(s'|s,a)\big[r + \gamma \max_{a'} q^*(s',a')\big]. \tag{19}$$

For finite MDP, the Bellman optimality equations for v^* (18) has a unique solution. It is actually a system of equation, one for each state with $|S|$ unknowns. If the dynamic P of the environment is known, than the MDP optimal policy can be inferred. Once v^* is known, the optimal policy could be easily determined by selecting the action that will lead to the next state with the higher value. If instead q_* is know, one could just find the action A_t that maximizes the q-value in the current state s. RL techniques are used when the dynamic of the system is not known or is too complex to be represented or simulated.

RL Taxonomy There is a variety of characteristics that a method for solving MDP can have. An algorithm can be *online* or *offline*. In the online case, the value-functions updates or the policy updates are performed after gathering a handful of samples from the environment. This procedure needs a constant interaction between the agent and the environment, since the update samples are not stored but are requested at will. An important advantage of these methods is that the learnt policies are very robust, since the agent is continuously communicating with the agent and can therefore correct its behaviour on the fly in case of drastic changes in the environment dynamics. In addition, the agent can finely control the amount of exploration and exploitation needed for a high quality learning. On the other hand, an offline algorithm separates the sampling task from the policy optimization task. A dataset of state transitions is sampled with a given policy and stored for later use. Then, the policy optimization is performed on the whole dataset. The main advantage of this technique is that the algorithm does not need constant access to the environment during the training phase, and the environment sampling can be formed just once for multiple run of the same algorithm. Since the optimization is not performed in real time, there is no time constraint on the policy search task, and therefore complex models can be built.
Another important distinction present in RL literature is the *single-objective / multi-objective* dichotomy [19]. A single-objective algorithm tries to maximize a single reward function. On the other hand, a multi-objective algorithm extends the reward function R such that it outputs a multidimensional value $r \in \mathbb{R}^n$. Multi-objective learning is useful when multiple conflicting metrics must be optimized. However, it's possible perform a *scalarization* of the multi-objective problem and have all the different metrics expressed as a single weighted sum.
The last important characterization of RL algorithms is based on which part of the learning problem is parametrized. *Value based* algorithms try to build a representation of the value functions. In this case, there is no explicit representation of the policy, but the main advantage is that is possible to link every state to the value that it has. The policy is than derived by finding the action that maximized the change of state. *Policy based* algorithms instead optimize directly the policy, without representing explicitly the value functions. The policy $\pi(a|s,\theta)$ is optimized and its parameters θ are update by gradient ascent on $\mathbb{E}\big[R_t\big]$. The main advantage of this approach is that this algorithms model directly the behaviour of the agent rather than retrieve it from a implicit representation encoded by the value function. This is optimal in the case of states encoding with a massive number of dimensions: rather that waste parameters in state representation, only the behaviour is modeled, having a higher sample efficiency. An *Actor-Critic* algorithm learns both the policy and the state value function. The term *actor* refers to the learnt policy that performs action selection, while the *critic* refers to the learnt value function. After picking each action, the evaluation of the new state is done by the critic, that will decide if the action proposed by the actor will have positive effects.

Fitted Q-Iteration Fitted Q-Iteration (FQI) [20] is an off-policy, value based algorithm. It is used to estimate the action-value function and derive a control policy from a batch of transitions sampled from the environment. The transitions are sampled with a given policy, whose exploration capabilities will have an effect on the quality of the estimates of the Q-function.
A transition is a four-tuples $\langle s_t, a_t, r_t, s_{t+1}\rangle$ such that:

Algorithm 1 **Fitted Q-Iteration (FQI)**

Inputs: a set of four-tuples $\mathbb{F}$ and a regression algorithm
Initialization:
Set i to 0
Let $\hat{Q}_i$ be the function equal to zero everywhere on $X \times U$
for $i < N$ **do**
    $i = i + 1$
    Build the training set $TS = \{(i^l, o^l), l = 1, ..., \#\mathbb{F}\}$ based on the function $\hat{Q}_{i-1}$ and on the full set of the four-tuples $\mathbb{F}$:

$$i^l = (s_t^l, a_t^l) \tag{21}$$

$$o^l = r_t^l + \gamma \max_{u \in U} \hat{Q}_{i-1}(s_{t+1}^l, a) \tag{22}$$

    Use the regression algorithm to induce from TS the function $\hat{Q}_i(s, a)$
end for
return the trained regressor

- s_t: starting state of the transition.
- a_t: action drawn from the exploratory policy.
- r_t: reward obtained by the agent after performing the action a_t in the state s_t.
- s_{t+1}: ending state reached after performing the action a_t in the state s_t.

Also, let Q_i be the action-value function computed on a time of horizon of i steps.
The idea behind FQI is to build at every training step i an approximation of Q_i from the transition dataset with a supervised learning method. Then, the same regressor is trained on this newly generated dataset, and this operation is repeated till a termination condition is met. The input of the regressor is the current state s_t and the action a_t, while the target is built plugging the previous approximation of the Q-function in the Bellman optimality equation. The control policy can be retrieved my maximizing the approximated Q function with respect to a.
The original work studies the convergence properties of the algorithm while using different classes of tree-base regression models. In this work the **XGBoost regressor** is used, due to its high scalability and efficiency.
XGBoost [21] is a tree-boosting algorithm that uses an ensemble of decision trees to perform regression on a dataset $\mathcal{D} = \{(x_i, y_i)\}$ using $\mathcal{K}$ additive decision trees to predict the output:

$$\hat{y}_i = \sum_{k=1} f_k(x_i) \qquad \text{for } f_k \in \mathcal{F}, \tag{20}$$

where $\mathcal{F}$ is the space of regression trees. In this type of trees, each leaf corresponds to a continuous score. One tree at a time, XGBoost trains the $\mathcal{K}$ decision trees in order to minimize a loss and greedily adding it to the ensemble. This process of using previously computed solutions is called *boosting*. A pseudo-code corresponding to the FQI algorithm is presented in Algorithm 1.

2.4. Related Works

PV power generation, battery degradation, and energy arbitrage are complex problems often studied individually. Sui et al. [22] study the problem of scheduling charge, discharge, and resting periods while using multiple batteries. The proposed scheduler has to keep the SoC of every battery over a given level, and at the same time, it has to minimize the degradation caused by high temperatures. It models two different characteristics of a Lithium-Ion battery: *rate capacity effect* and *recovery effect*. Due to the former, a battery shows a smaller overall capacity when discharged at high currents, while the latter influences the battery voltage recovery after a continuous discharge process. The scheduler takes advantage of these two effects and extends the battery life. This work considers fixed charge/discharge currents. While this approach simplifies the control problem, it does not allow the scheduler to choose between different charging or discharging profiles that could achieve the same performances with lower effects on the degradation. Another shortcoming of this work is that SoH modeling is influenced only by temperature, and other important factors such as DoD, SoC, and current rate are not considered. Moreover, no economic considerations are done w.r.t. SoH, and the objective of the scheduler is just to use for as long as possible a battery while avoiding cycles that generate short-term high degradation.
Huang et al. [23] consider a Energy Storage System (ESS) that manages energy produced from renewable sources. Solar and wind energy are characterized by periodic patterns that can be predicted by taking into

account meteorological data. This work ties the decision process by predicting the implants' energy availability produced. The system should follow the energy production profile to store the most energy possible and then sell it when the market conditions are profitable. The controller is designed with an economic perspective: the objective is to maximize the profit by selling energy while keeping into account the operational constraints of the ESS. This work, however, does not make considerations about the SoH of the battery packs that compose the ESS, and therefore does not consider the effect of the degradation on the profit. The ESS is assumed to have a maximum capacity fixed over time, and it is not taken into account the economic effects of purchases of new accumulation systems.

We remark that RL has been used in multiple real-world settings such as finance [], autonomous driving [24], environmental control of dams [25], social networks intention discovery [26]. Regarding its application in the smart grid setting only a few work are worth mentioning. Cao et al. [27] design a noisy network deep reinforcement learning controller that performs energy arbitrage. The objective is to generate profit by storing or releasing energy from an accumulation system, which is bought only to perform arbitration. There is no component in the considered system that produces energy, and therefore only exchanges with the electric grid are allowed. This technique considers past electric market prices history and can make a prediction for the next 24 hours. Then, based on the prediction, the controller decides which interaction with the grid is most profitable. The peculiarity of this work is that it considers the effects that battery degradation has only in the profit estimates. The main drawback of this approach is that profit is computed on short-term (*i.e.,* one-week) periods and, therefore, does not consider profit for the long time horizon.

Kell et al. [28] use a deep reinforcement learning controller to regulate energy usage in homes with photovoltaic panels and an accumulation system installed. This technique allows fine-grained control of the current to which a battery is subject, allowing efficient and precise driving. The main objective of the work is to find the correct battery size for a given household. However, the technique is based on the weak assumption that no relevant battery degradation happens in one year, and the controller needs to be re-trained every time a new battery is considered.

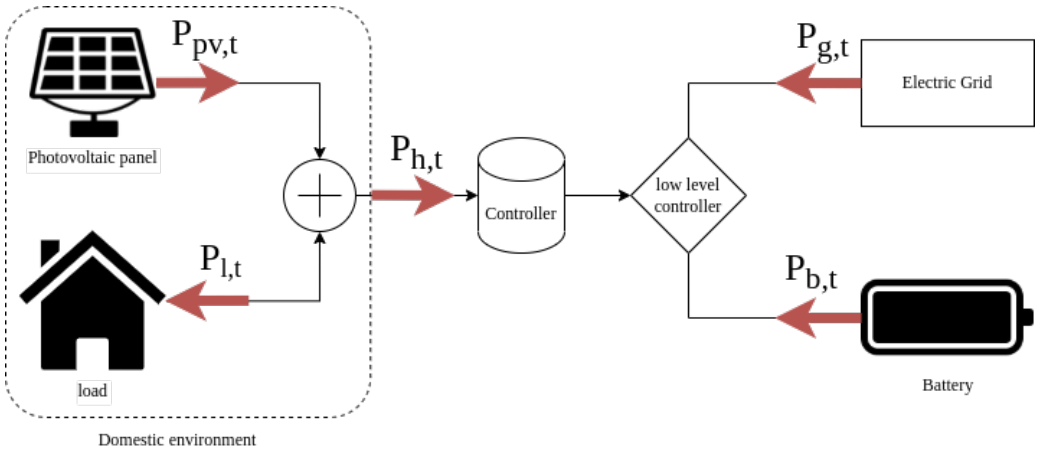


Figure 4: The domestic environment is composed by the residence of an user and a photo-voltaic panel. The controller can read the net power consumption $P_{h,t}$ and decide how much power will be routed to the battery and to the electric grid.

3. Problem Formulation

Consider the context described in Figure 4. It considers a domestic environment where a photo-voltaic panel (PV) and and a consumer (i.e., the house) are interacting. The PV produces energy in a periodic way, following the cycle of the sun, while the habitation will consume energy following the routine of its owners. The domestic system as a whole will alternate periods in which it has an energy surplus with others with energy shortage.
A controller links the domestic environment with an accumulation system and the public electric grid. The accumulation system is made of a Lithium-ion batteries pack used to store energy produced in excess. This energy can be retrieved at a later time in order to meet the domestic environment demand. The battery is freshly installed and it has been bought at the price c_b. The electric grid can supply or receive at energy respectively at the market prices c_g and p_g. The controller monitors these subsystems with a fixed control period Δt for a predetermined number of control steps $\mathcal{T}$. It reacts to events happening in the domestic system by interacting with the accumulation system and the electric grid.
In order to achieve this, the controller is able measure the state of each system at every time step t. In fact, it is able to measure the power generated by the PV $P_{PV,t}$ and the net power $P_{h,t}$ outputted by the whole domestic system, defined as:

$$P_{h,t} = P_{l,t} - P_{PV,t} \qquad \forall t = 0, \dots, \mathcal{T}. \tag{23}$$

For what concerns the accumulation system, the controller can measure how much power $P_{b,t}$ is stored or retrieved. It can also monitor several physical properties of the battery pack, such as its SoC σ_t, the battery temperature $T_{b,t}$, the environment temperature $T_{env,t}$, the DoD of the current cycle δ_t, and the degradation of the battery D_t. The controller can send or request power $P_{g,t}$ from the electric grid generating an economic transaction.

Controlled Variables The task of the controller is to measure every time step t the incoming power $P_{h,t}$ and decide which percentage $\alpha_t \in [0, 1]$ store (retrieve) in (from) the battery. The rest of the power is directed to the electric grid. In formulas:

$$P_{b,t} = \alpha_t P_{h,t} \qquad \forall t = 0, \dots, \mathcal{T}, \tag{24a}$$
$$P_{g,t} = (1 - \alpha_t) P_{h,t} \qquad \forall t = 0, \dots, \mathcal{T}, \tag{24b}$$
$$E_{g,t} = P_{g,t} \Delta t \qquad \forall t = 0, \dots, \mathcal{T}. \tag{24c}$$

Note that, following the notation in Fig. 4, if $P_{b,t}$ is **positive**, than the battery is **discharging**. At the same time $P_{g,t}$ is **positive** if the power is **requested** from the grid. Eq. (24c) computes the actual energy exchanged with the grid, and this quantity will be used to compute the economic gain or loss occurred while interacting with the electric grid.

Constraints While operating, the controller should comply with some battery capacity and degradation constraints. The SoC follows the dynamics expressed in Equation (2), and at every time step must hold:

$$0 \leq \sigma_t \leq 1 \qquad \forall t = 0, \dots, \mathcal{T}, \tag{25}$$

In this setting, a low level controller is assumed to be present behind the high level controller. It protects the battery from overcharging or excessive draining, by actuating the high level action such that Equation (25) holds. The controller has also to keep track of the degradation that the battery is subject to:

$$D_t = f(t, D_{t-1}, \bar{\theta}_{SoH}) \qquad \forall t = 0, \dots, \mathcal{T}, \tag{26a}$$

$$0 \leq D_t \leq D_{max} \qquad \forall t = 0, \dots, \mathcal{T}, \tag{26b}$$

$$\Delta D = D_{\mathcal{T}} - D_0 \leq D_{max}, \tag{26c}$$

The control episode is run for $\mathcal{T}$ steps, or when the degradation reaches a threshold D_{max}. The degradation is computed through Equation (26a), where f is the degradation dynamics. It is a monotonous function that depends on the current time step, the last measured degradation D_{t-1} and some parameters $\bar{\theta}_{SoH}$ that characterize the battery. At the end of the episode, the battery has incurred in a degradation of ΔD.

Objective The objective of this work is to find the sequence of actions that reduce as much as possible the amount on money needed to maintain the system, and to generate a profit while exchanging electric energy with the grid. The objective can be divided in two parts: battery cost and energy trading. The former takes into account how much money is lost due to degradation, and it grows linearly with said degradation. The latter refers to the profit made by interacting with the electric grid. This two objective are conflicting by nature, since a more aggressive use of the battery could generate favourable trades with the electric network, but it will also consume the battery faster. This can be expressed as:

$$\max_{\alpha_0, \dots, \alpha_{\mathcal{T}}} -\frac{\Delta D}{D_{max}} c_b + \sum_{t=0}^{\mathcal{T}} \Big[p_g E_{g,t} \mathbb{1}_{\mathbb{R}^-}(P_{g,t}) - c_g E_{g,t} \mathbb{1}_{\mathbb{R}^+}(P_{g,t}) \Big], \tag{27}$$

where $\mathbb{1}_{\cdot}(\cdot)$ is the indicator functions, defined as:

$$\mathbb{1}_{\mathcal{A}}(x) = \begin{cases} 1 & \text{if } x \in \mathcal{A} \\ 0 & \text{otherwise} \end{cases}. \tag{28}$$

4. Algorithm

The problem above mentioned in Section 3 is directly influenced by the degradation model expressed in Equation (26). In this work, the model already discussed in Section 2 is used, since it allows to take into account the degradation effects of SoC, DoD and temperature. Unfortunely, it has the shortcoming of not considering the degradation caused by charges or discharges performed at high C-rates. This effect is however partially captured by taking into account the effects of high temperature cycling.
In what follows, we report a version of Equation (7) and (8) are reported, depending on the current time step t:

$$f_{cal,t} = f(t, \bar{\sigma}_t, \bar{\delta}_t, \bar{T}_t) \qquad \forall t = 0, \dots, \mathcal{T}, \tag{29a}$$

$$f_{cyc,t} = \sum_{i=0}^{N_t} n_{i,t} f_c(\delta_{i,t}, \sigma_{i,t}, T_{i,t}) \qquad \forall t = 0, \dots, \mathcal{T}, \tag{29b}$$

$$f_{d,t} = f_{cal,t} + f_{cyc,t} \qquad \forall t = 0, \dots, \mathcal{T}, \tag{29c}$$

$$D_t = 1 - \alpha_{sei} e^{-\beta_{sei} f_{d,t}} - (1 - \alpha_{sei}) e^{-f_{d,t}} \qquad \forall t = 0, \dots, \mathcal{T}. \tag{29d}$$

This degradation model uses the rainflow algorithm to quantify cycles in the battery SoC profile $\sigma_{0:t}$. Its output is then fed to Equation (29). The rainflow algorithm is called at every time step, together with the update of the system dynamics equation. These repeated calls causes a massive computation overhead, since rainflow has no memory of past runs, and therefore multiple computation are repeated. This problem is caused by the offline nature of the algorithm.
To have a more scalable algorithm, a new technique called *streamflow* has been developed. It performs cycle counting in an online manner by considering one sample at a time and following the rainflow rules in Section 2. When a new sample is added to the profile:

1. If the charging direction of the battery has changed, a new cycle is opened;
2. If the charging direction is the same as before:
   (a) if there is a cycle falling from above between the actual sample and the expected end of the current charge, assign the sample to that cycles;
   (b) otherwise, the sample is assigned to the cycle with highest or lowest end, depending if the current cycle is charging or discharging;
3. The mean, range, start and end of the current cycle are updated.

Rainflow uses the whole history in order to decide when a cycle opens and closes, but this information is not available to streamflow. Therefore, for each sample, the following approximation of the closing profile value is performed:

$$\sigma_{end} = \begin{cases} \sigma_t + \frac{1-\sigma_t}{2} & \text{if } (\sigma_t - \sigma_{t-1}) > 0 \\ \sigma_t - \frac{\sigma_t}{2} & \text{otherwise} \end{cases} \qquad \forall t = 0, \dots, \mathcal{T}. \tag{30}$$

We remark that Equation (30) tells that the expected end of the current charge is at half the distance from the SoC extreme point towards with the cycle is going. The main advantage of this approach is that cycling information is updated one sample at a time, and it can be done with a more amenable time complexity.
Since this degradation model depends on variables such as DoD and temperature, the formulation is extended with equations that model their dynamics. The DoD dynamics can be expressed as follows:

$$\delta_t = \begin{cases} \delta_{t-1} + |\sigma_t - \sigma_{t-1}| & \text{if } (\sigma_t - \sigma_{t-1})(\sigma_{t-1} - \sigma_{t-2}) > 0 \\ |\sigma_t - \sigma_{t-1}| & \text{otherwise} \end{cases} \qquad \forall t = 0, \dots, \mathcal{T}. \tag{31}$$

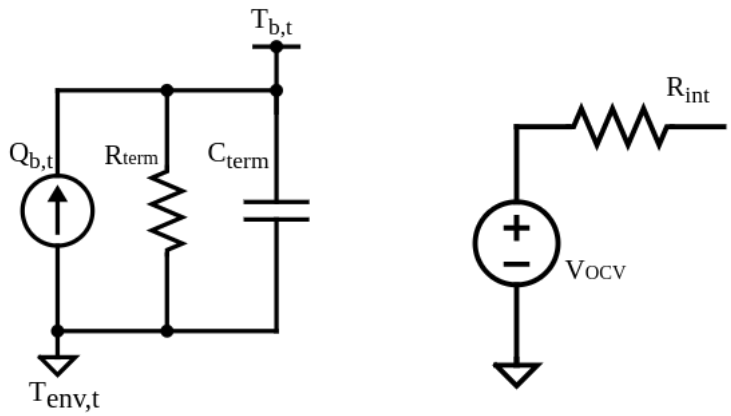
The DoD is defined as the absolute value of the SoC excursion of the current charge. Equation (31) conveys this concept with a differential formula: if the SoC is changing in the same direction of the last time step, than the DoD is growing, otherwise it is set to its initial value.
The thermal model defines how the battery temperature $T_{b,t}$ changes over time. In this context, the temperature behaviour is controlled by the heat dissipated due to the Joule effect during charges or discharges:

$$Q_{b,t} = I_{b,t}^2 R_{int}, \tag{32}$$

where R_{int} is the internal electric resistance of the battery. This effect is a consequence of modeling the battery like a real generator that exhibits resistive behaviours when a current passes through it. The temperature dynamics is modeled with the thermal circuit in Figure 5a:

$$L(s) = \frac{R_{term}}{R_{term} C_{term} s + 1}, \tag{33a}$$

$$T_{b,t} = \frac{Q_{b,t} R_{term} \Delta t + T_{b,t-1} R_{term} C_{term} + T_{env,t} \Delta t}{R_{term} C_{term} + \Delta t} \qquad \forall t = 0, \dots, \mathcal{T}. \tag{33b}$$



(a) (b)

Figure 5: Thermal (a) and electric (b) model of the battery.

Intuitively, Equation (33a) is the Laplacian transfer function, and it described how the heat exchanges happen between the battery and the surrounding environment. It is worth mentioning that the battery is stored at a temperature $T_{env,t}$. Moreover, Equation (33b) is the corresponding anti-Laplacian, and it describes how the temperature of the battery changes through time.

4.1. RL algorithm

The problem at hand is a sequential decision problem: the controller has to find the best sequence of actions that will maximize the objective in Equation (27). It is worth noting that $P_{l,0:\mathcal{T}}$ and $P_{PV,0:\mathcal{T}}$ are signals not known in advance: their values are in fact measured every control period Δt. It is not possible to perform action scheduling, since future events are not predetermined but depends on highly stochastic events such as the cycle of the sun or the consumption events in the house. This setting can be formalized as a *Markov Decision Process* $\mathcal{M}$. In what follows, the elements defining the MDP for the analysed problem are described.

State The state vector $s \in \mathcal{S}$ is defined as follows:

$$s = (\sigma_t, T_{b,t}, \delta_t, I_t^{req}, P_{PV,t}, \cos(\varphi_{d,t}), \sin(\varphi_{d,t}), \cos(\varphi_{y,t}), \sin(\varphi_{y,t})), \tag{34}$$

where t is the current time step, σ_t, $T_{b,t}$, and δ_t are the SoC, the battery temperature and the DoD, respectively, and are used to keep track of the state of the battery. These variables are also relevant since they have a direct impact on the computation of the degradation. Moreover, I_t^{req} is the maximum C-rate that the battery would be subjected to if all the net power $P_{h,t}$ would be directed to the battery. The use of C-rate is motivated by the fact that the C-rate values have the same meaning for different sized batteries, and, therefore, this state encoding can be used in multiple situations. $P_{PV,t}$ is the power generated by the PV, and is used to have a rough prediction on the future sun availability: in fact, if during the day this value is very low, one can assume that the day is cloudy or rainy and no future power production is expected. The last four components are used to map the current time into an encoding that is able to express a similarity measure between difference periods of the day/year [29]. This problem is characterized by two types of periodicity: the day-night periodicity and the seasonal periodicity. This is done by mapping the current time to a position on the unit circumference. φ_d is the angular position for the time of the day, formally:

$$\varphi_d = \frac{2\pi\tau_d}{\mathcal{T}_d}, \tag{35}$$

where $\mathcal{T}_d$ are the seconds in a day and $\tau_d \in [0, \mathcal{T}_d]$ is the current second of the day, and φ_y is the angular position for the time of the year, formally:

$$\varphi_y = \frac{2\pi\tau_y}{\mathcal{T}_y}, \tag{36}$$

where $\mathcal{T}_y$ are the seconds in a year and $\tau_y \in [0, \mathcal{T}_y]$ is the current second of the year.
SoH and the number of equivalent cycles were included in the state definition in the first iterations of this work. In fact, the SoH value allows the agent to understand at which point of the degradation curve the battery

is at. This can give an important insight on the amount of future degradation expected, since the majority of the loss of capacity is placed at the beginning or end of the battery life. The number of equivalent cycles is the cumulative value of the DoD of each charge and discharge, and expresses how much the battery was actually used by the agent. However, this state variables are highly correlated with time, and the algorithm was over-fitting their values ignoring all the other features.

Action The actions formulation is the same explained in Section 3. The agent can select eleven discrete actions $\alpha_t \in \{0.0, 0.1, \dots, 1.0\}$. At every time step, the controller has to decide the percentage α_t of the net power $P_{h,t}$ that will be directed to the battery. The remaining power is sent to the electric grid. The effects of the actions are reflected instantly in the next state s_{t+1}. The low level controller checks the feasibility of the action proposed of the agent: if the SoC constraint would be violated, it actuates the biggest admissible action. It may occur that multiple actions may have the same identical effect because they have been corrected into the same actuated action. It's important to note that the agent is not punished for selecting unfeasible actions.

Reward The reward function is defined as follows:

$$r_t = -\frac{f_{d,t} - f_{d,t-1}}{f_d^{max}} c_b + p_g E_{g,t} \mathbb{1}_{\mathbb{R}^-}(P_{g,t}) - c_g E_{g,t} \mathbb{1}_{\mathbb{R}^+}(P_{g,t}). \tag{37}$$

Intuitively, Equation (37) takes into account the profit made by the agent by exchanging energy with the electric grid and the amortization of the battery value during the operational period. The energy exchange with the grid generates a profit of p_g per kWh sold, otherwise the agents incurs in a loss of c_g. At the same time, the battery degradation has increased due to passive factors such as time passing, or due to active factors such as active cycling.
The battery value is amortized by considering the variation in *linear degradation* f_d, rather then SoH. The difference in SoH was not used because it generated agents that avoided as much as possible battery cycling. This conservative behaviour can be explained by the massive loss in degradation (and therefore in value) that a battery experience at the beginning of its life, since its degradation curve is very steep while it is still fresh. On the other hand, by using the linear degradation, the reward is distributed more fairly on the whole time period, but the agents is still able to understand how much of an impact an action had on the degradation. This allowed the training of agents that truly maximized the long term profit and that were able to make a trade-off in profit at the beginning of the battery life.
The whole battery value is amortized with respect to the f_d^{max}, the maximum linear degradation value that corresponds to the maximum degradation D_{max}. Formally:

$$D(\mathbf{f_d^{max}}) = 1 - \alpha_{sei} e^{-\beta_{sei} \mathbf{f_d^{max}}} - (1 - \alpha_{sei}) e^{-\mathbf{f_d^{max}}} = D_{max}. \tag{38}$$

Notice that Equation (38) is not invertible in closed form, due to the presence of two exponential with different exponents. The value f_d^{max} therefore has be estimated with the bisection algorithm [30]. The γ factor used in this problem as been set to 1, since it is a problem with a finite number of steps $\mathcal{T}$. This has also the advantage to create a very farsighted agent that will be able to optimally use the battery on the whole time horizon, and will avoid to maximize the current profit at the expanse of future gains.

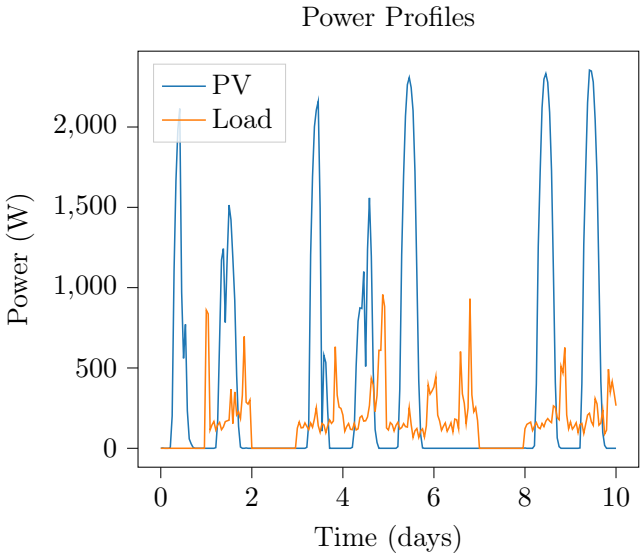


Figure 6: Power profiles used in the Gym environment.

5. Experimental Results

5.1. Experimental setting

In order to test the solution proposed in Section 4, an online simulator which implements the OpenAI Gym standard framework [31] is implemented. The simulator implements all the system dynamics and degradation formulas discussed in Sections 3 and 4. A core part of the simulation is the power signal generated by the PV $P_{PV,0:\mathcal{T}}$ and the auto-consumption profile $P_{l,0:\mathcal{T}}$, both generated from real data. The PV profile is generated starting from a pool of sixteen year-long profiles, gathered from different power plants. These profiles are normalized with respect to the maximum power that the solar plant generated during its operational life. This opens the possibility of scaling the power generation profile by any multiplicative factor, so that it can be coupled with any auto-consumption profile. The multiplication factor controls the energy balance of the domestic system, allowing to control how much the domestic system is energy restricted.
The load profile is generated from a pool of 398 annual profiles, with peak consumption of $2 - 3$ kWh.
Figure 6 shows 10 spring days of both a PV and auto-consumption profile. In the PV profile, three different energy production patterns are present: clear days, characterized by a bell-shaped curve, cloudy days, with an irregular shape, or dark days where no power was produced. The load profile is characterised by a low constant consumption of 200 W with peaks during mornings and evenings. The flat lines in the profiles can be attributed to momentary malfunctions in the monitoring system. Energy production and consumption periods are not aligned, but show up in complementary parts of the day. The control task will have to understand which is the best policy that is able to adapt to the asynchronicity of these processes.
Every time an episode starts a new net profile $P_{h,0:\mathcal{T}}$ is created, combining a PV and load profile. The PV profile is scaled by the peak consumption of the load profile and is subject to data augmentation: each day is drawn from a pool of 7 days from a sliding window centered in the current day. The original data is sampled every hour, and therefore the control period is $\Delta t = 3600$ s. It is important to note that even though the Gym environment has access to the whole power profile, this information is given to the agent one sample at a time at the appropriate moment.
The parameters of the simulation are reported in Table 1. Every episode is run for **8 years** or when the maximum degradation threshold D_{max} is met. For what concerns the thermal model in Equation (33b), the ambient temperature T_{env}, t is considered fixed at 25°C. The same holds for the energy prices exposed by the electric grid: p_g and c_g are stationary and are respectively fixed at 15 cents/kWh and 5 cents/kWh. The original parameters of the degradation model proposed in [13] where used.
Every episode considers a different battery, whose capacity is dimensioned for a mean up-time of 1000 hours in a whole year:

$$C_b = \frac{1000 P_{max}}{365 V_{max}}, \tag{39}$$

where C_b is the battery capacity expressed in Ah, P_{max} the maximum power generated from the PV and V_{max} is the nominal voltage of the battery. The battery cost is fixed for any episode and is proportional to the battery

Table 1: Parameters of the Simulator.

Thermal Model			
R_{term}	0.37 °C/W	C_{term}	1.7e3 J/K
R_{int}	5e-3 Ω	T_{env}	25 °C
Economic Parameters			
$c_{b,unit}$	375 €/kWh	p_g	0.15 €/kWh
c_g	0.05 €/kWh		
Battery Parameters			
V_{max}	48 V	D_{max}	0.2
V_{OCV} Parameters			
a_0	38.83	a_1	7.7
a_2	-7.7	a_3	9.17
b_0	-3.51	b_1	-46

capacity. Every battery is bought at a unitary cost of $c_{b,unit} = 375$ €/ kWh, and it is computed such that the expected 8-years profit of a policy that uses always the battery is the same same of one that never uses it. The expectation was computed on the same 10 profiles by using the bisection method. The reason for this operation is that, by leveling off this two policies, the agent has to learn to truly optimize the use of the battery and the energy exchanges, avoiding these two simple cases.
Training and testing were performed on a virtualized *Ubuntu 20.04 LTS server*, on a *64x Single Core Intel Xeon (Skylake IBRS)* and *32 GB of dual channel RAM.*

5.2. Training

The agent was trained with Fitted Q-Iteration (FQI). Since it is an offline algorithm, a dataset of transitions needs to be generated. A standard approach in the dataset generation would consists on sampling uniformly the state space and than apply a random action. This would allow maximum exploration and the agent would learn the reward signal on the whole state space. However, the degradation model depends on the whole battery history, and therefore is not possible to compute a transition from a random state.
On the contrary, 100 episodes were created and explored with a uniformly random policy. A total of 7 million state transitions where sampled in one hour.
Fitted Q-Iteration (FQI) was then run for 200 iterations by using an XGBoost regressor. The regressor was trained to minimize the mean squared error with the following hyperparameters:

- *number of trees*: 1100;
- *maximum depth*: 8;
- *colsample_bytree*: 0.8;
- *subsample*: 0.8;
- *alpha*: 0.1;
- *lambda*: 0;
- *tree_method*: "hist";

5.3. Testing

Agent Performance The agent was tested against 4 different Key Performance Indicators (KPI):

- *Profit*: the value of the objective function.
- *Battery Cost*: the first component of the objective function, expresses how much value of the battery was lost while cycling.
- *Energy Profit*: the second component of the objective function, it is the profit made by exchanging energy with the electric grid.
- *Degradation*: degradation D_t that the battery was subject to.

The KPIs where sampled every 30 days from 10 different episodes and then averaged.
The performance of the agent is compared with 3 different baselines:

- *OnlyGrid*: The actions of this baseline are always set to 0. No power falls on the battery, and therefore only the calendar ageing impacts on the degradation. It has the main advantage of preserving the battery, but the energy exchanges with the electric grid are always disadvantageous, since $c_g << p_g$.

Table 2: Average KPI values after 8 years.

	Profit	Battery Cost	Energy Profit	Degradation
Agent	-2295.32	-1680.54	-614.78	0.1245
SoC20-80	-2454.99	-2144.69	-310.30	0.1589
OnlyBattery	-2365.93	-2141.18	-224.74	0.1586
OnlyGrid	-2354.15	-1531.66	-822.49	0.1135

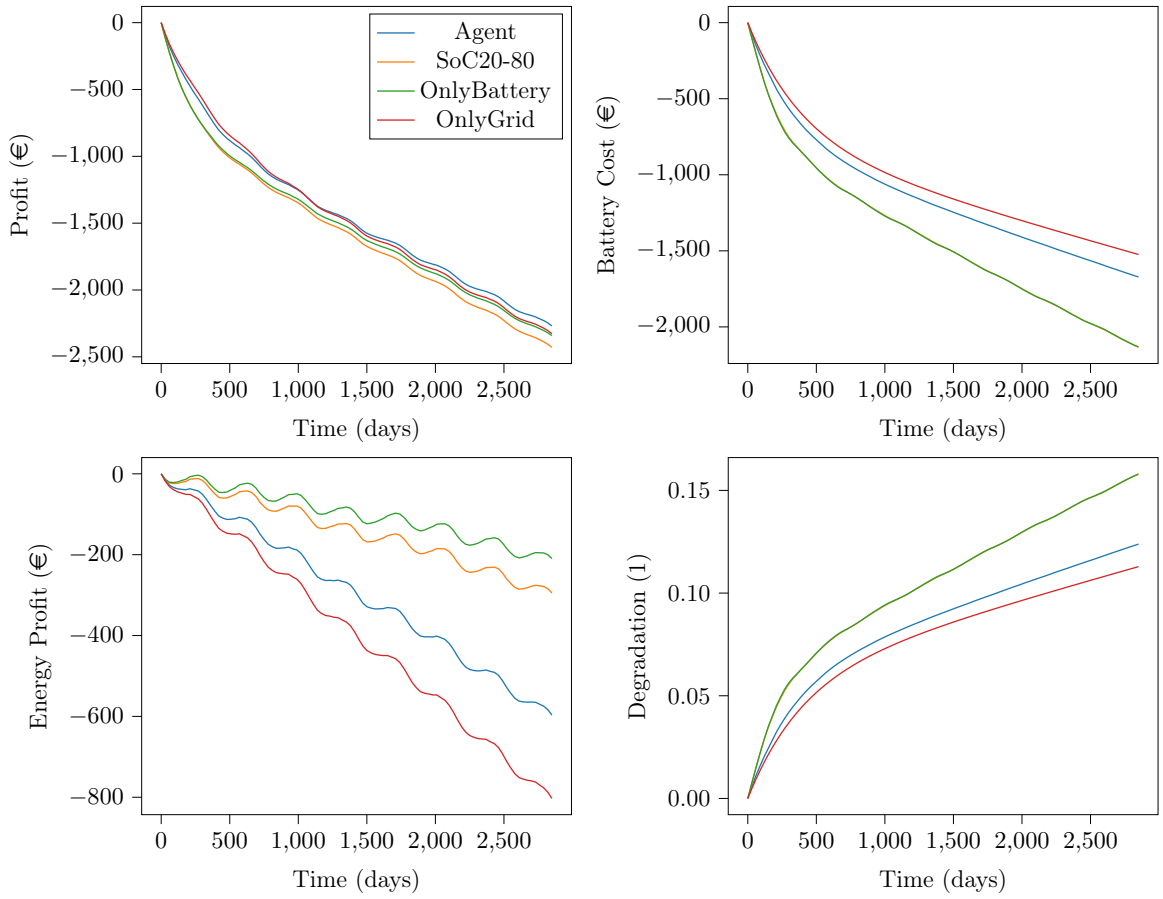


Figure 7: Mean absolute performance on 10 episodes of the policies in 8 years. The best agent for each category is the higher in the graph, with the exception of Degradation.

- *OnlyBattery*: This baseline always uses the battery. Energy exchanges with the grid happen only when the battery is completely empty or full. This policy has a dramatic effect on SoH, but is able to store energy for later use and avoids buying power from the grid.
- *Soc20-80*: This baseline keeps the SoC between 0.2 and 0.8. This is the state of the art control policy [32], since very low or very high SoC values are correlated to high degradation. It can be considered a mix of the two previous strategies.

Since the cost of the battery was balanced, OnlyGrid and OnlyBattery perform in a similar way on an 8 year span and the main difference in their tests results is due to the inherent variance present in the system dynamics. However, they reach the same profit by operating one on the battery degradation, the other on the energy exchange, and therefore it is still useful to consider them since they both can give an insight on the agent results.

Figure 7 presents the performance of the agents and the three baselines are considered. All the policies are not able to generate a positive profit. This may be caused by the stationary energy prices hypothesis, that gives very little room for optimization since it is not possible to take advantage of price fluctuations. In fact, the energy loss incurred when selling and then buying back the energy is always fixed at −10 cents/kWh. In Figure 8, the same data is shown but with a different perspective. The baseline SoC20-80 is considered as a base and the difference in KPI value for the other policies are plotted. Unexpectedly, this baseline performs way worse then the other policies, since it degrades heavily the battery like the OnlyBattery baseline without being able to have advantageous energy exchanges. These low performances may be explained by the use of

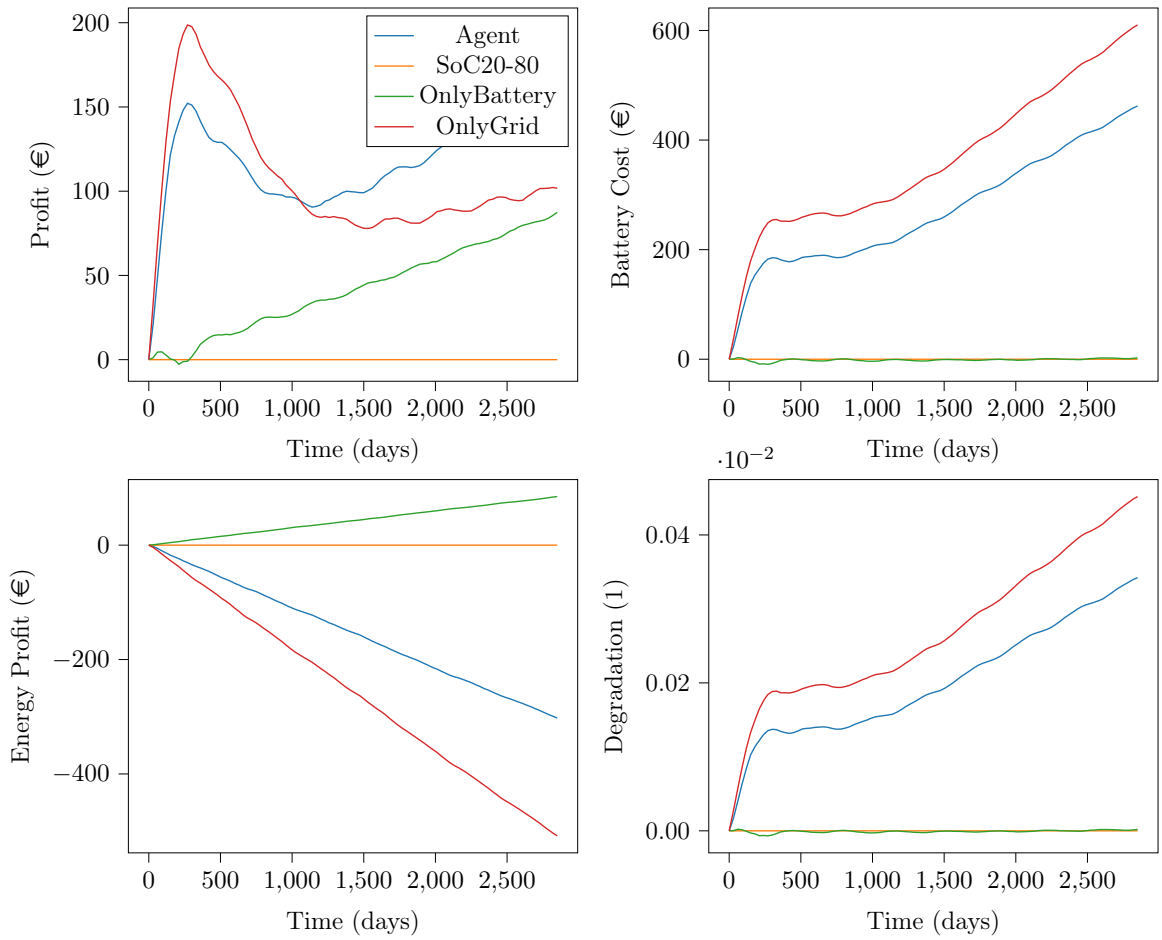


Figure 8: Mean performance on 10 episodes in 8 years, evaluated with respect to the baseline Soc20-80. The best agent for each category is the higher on the graph.

fixed energy costs or disadvantageous parameters in the degradation model. On the other hand, the trained agent performs greatly, achieving almost 150€ in profit with respect to the reference point and almost 100€ more than OnlyGrid, as shown in Table 2. It is important to note that the agent was able to reach this result without excelling in both battery cost or energy profit. This means that its possible to find a policy that is able to exploit the degradation dynamic without fixing its action to a specific value. Even though the agent had excellent performances on the overall period, the agent start to perform better than all the other baselines after 3 years: of early high profits is not alarming, since the investment has to pay-off on the whole 8 years period.
Fig. 9 shows which are the states most visited by the agent and the baseline Soc20-80. The baseline tries to limit its SoC level, but sometimes the threshold are exceed since the whole power generated or requested by the domestic environment is actuated. Most of the states are concentrated in the principal diagonal and around the 20% and 80% SoC level. The agent, on the other hand, spends most of the time at low SoC. At the same time, when the agent is cycling at high SoC, it is done while performing low DoD cycles. In this way the degradation caused by high SoCs is mitigated by performing small cycles, reducing the degradation apported by the DoD.

Agent Behaviour The agent has learned a complex policy, and the use of XGBoost has allowed the representation of a highly non-linear behaviour. Since the FQI algorithm is a value-based technique, the regressors have learned the action-value function, and therefore is possible to extract its value and understand the agent behaviour. Q-value can be used to study which are the most convenient states and why a particular action has been actuated. Due to the use of a unitary discount factor γ, it has not a precise economic meaning, but a high value can be linked to future profits while a negative value corresponds to an expected monetary loss.
Fig. 10 shows the trend of the Q-value during April and on a whole profile. The Q-value is characterized by two main 24 hours and 365 days periodic components. The high frequency component peaks every day during maximal energy production, since the presence of the sun will lead to a profit in energy exchange. Sunny days are characterized by higher peaks than rainy days. On the other hand, the annual periodicity follows the seasonal cycle, since the agent is able to generate more revenue during longer days.
The agent is able to achieve the best profit by avoiding high stress conditions such as high values of SoC and

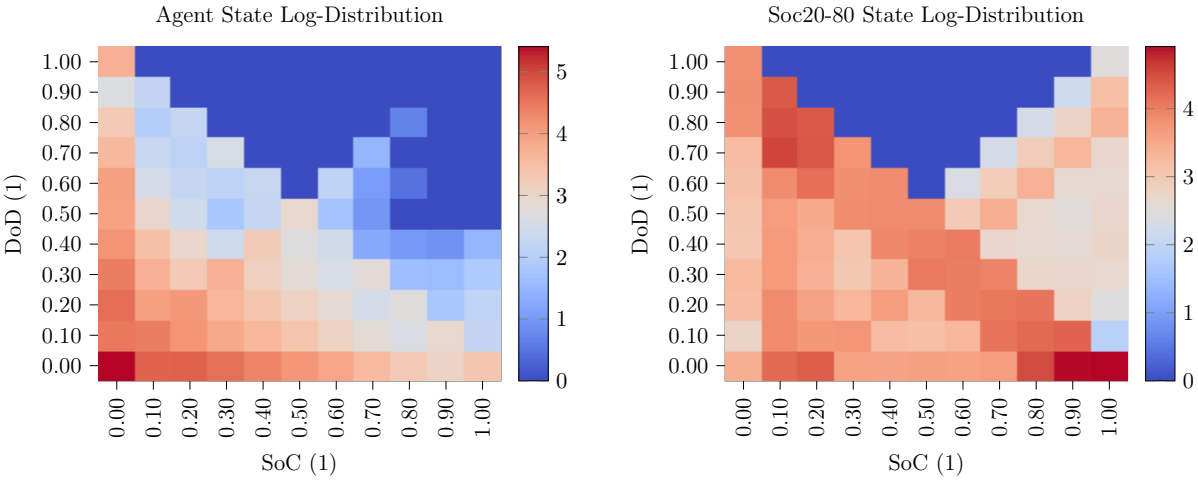


Figure 9: SoC-DoD log-distribution.

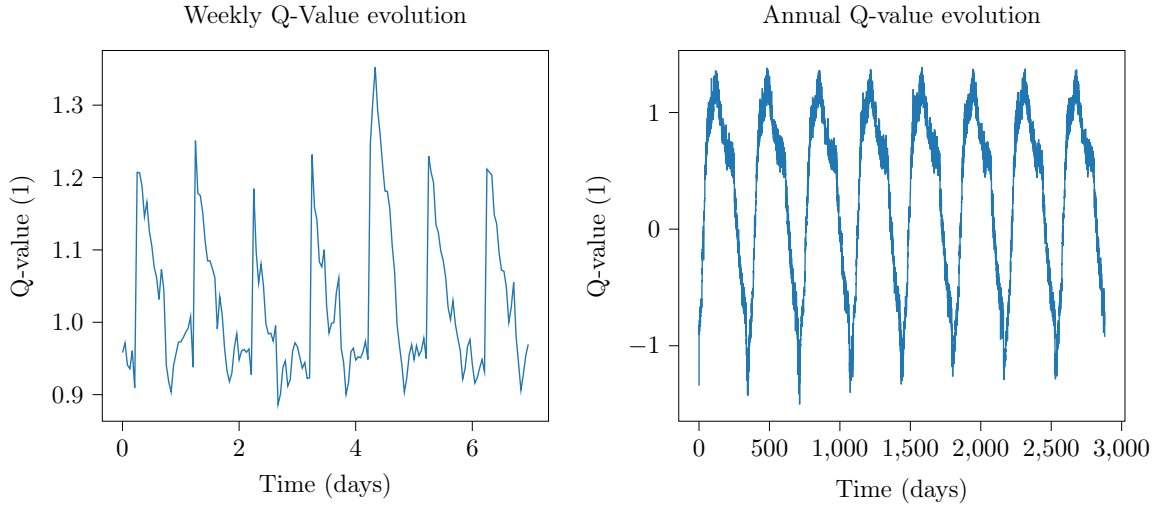


Figure 10: Trend of the Q-value over the whole 8 years period.

DoD, by actuating intermittent low power charges. Figure 11 shows the relative frequency of picking an action at a given SoC value. The heatmaps were created by counting how many times an action is picked at a given SoC level in the test set and then normalized with respect to the total number of actions done at said SoC. If the battery is charging, the agent avoids battery cycling most of the time (up to 50%), and power is accepted only in small quantities. On the other hand, while discharging the agent picks high actions, in order to discharge as fast as possible and lessen the degradation caused by high SoC cycling.
Figure 12 represents Q-value and Action functions trend. C-rate is crucial to determine the best action, since the action distribution is heavily influenced by its sign. In fact, it reinforces the consideration done before that states that the agent avoid as much as possible charging and as soon as it is possible to discharge the battery, it will to it actuating the biggest action. SoC has more effect on the Q-value, with larger peaks at lower values, since less degradation can happen in that range.

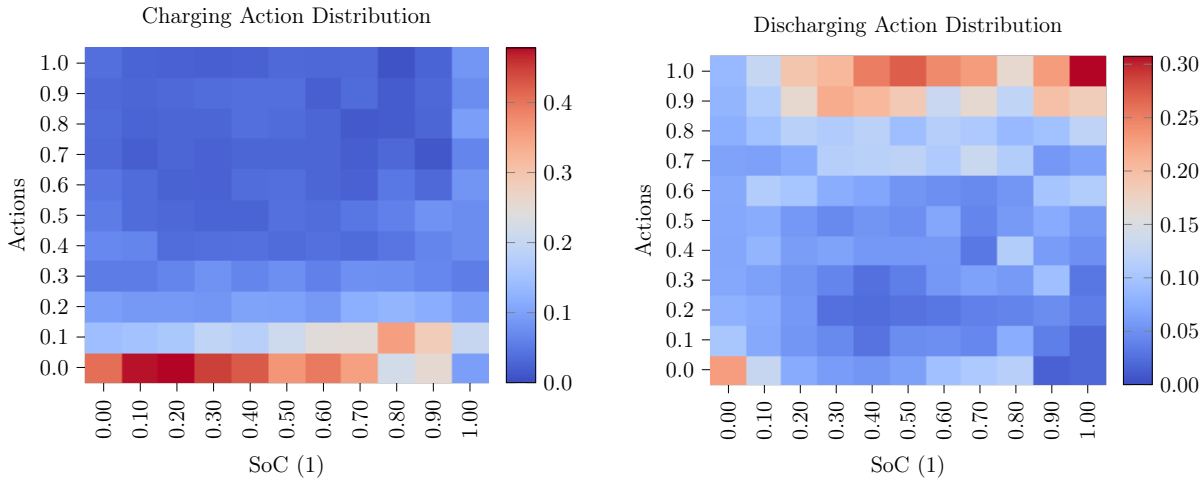


Figure 11: Q-value and actions distribution with respect to SoC and C-rate.

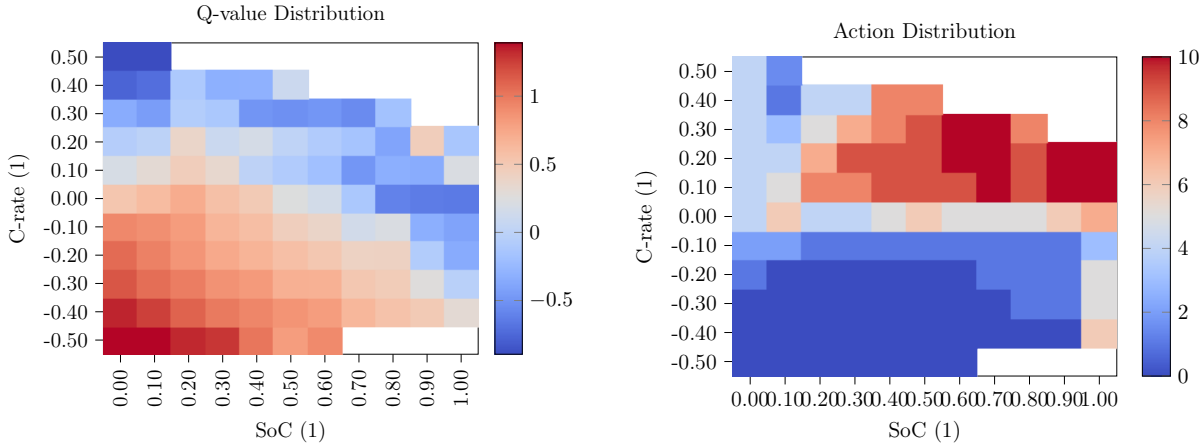


Figure 12: Q-value and actions distribution with respect to SoC and C-rate. A white cell represents states never visited by the agent.

6. Conclusions

Photovoltaic panels are used in residential environments to produce cheap and clean energy, lowering electricity costs and increasing energy independence. Profit is generated by meeting the domestic demand, thus avoiding expansive energy purchases, and by performing energy arbitration. The main difficulties in managing such systems are caused by the unpredictable nature of solar energy production and by the asynchronicity between energy production and consumption. To alleviate these limitations, an accumulation system is used, where energy in excess can be stored for later use.
However, Lithium-ion batteries, which are the most used type of storage systems for such a task, are characterised by a process degradation influenced both by environmental factors and dynamic loading. Indeed, different cycling conditions will result in a longer or smaller life, having drastic effect on the sustainability of the investment. Previous work considered domestic energy production by disregarding the battery degradation, or taking into account the battery degradation but having with no independent energy production.
This work design a RL controller trained with Fitted Q-Iteration (FQI) and the ensemble tree regressor XGBoost, by considering a degradation model that allows to compute instantaneous SoH loss. The objective is to maximize the long term profit while exchanging energy with the electric grid and by amortizing the battery cost on the whole period accordingly to its use. The algorithm proposed outperforms the state-of-the-art techniques by up to 15%, with a control policy that keeps SoC values as low as possibles with slow low-powered charges and fast discharges. The controller achieves great generalization, since it is able to operate different battery capacities without a dedicated training and it has been designed on multiple energy consumption routines.

Future works The problem at hand is a combination of multiple complex sub-problems, such as solar energy prediction, electricity prices prediction, energy arbitration and degradation modeling. A series of simplifying hypothesis were performed. Future works will have to:

- consider non-stationary energy prices. Efficient energy arbitration is possible to achieve when there are prices fluctuation that can be exploited to perform a profit. With fixed prices, the agent is only able to limit its net loss, since every energy exchange with the grid does not generated a profit. In this case, the agent is only able to store energy in order to meet future demand and avoid energy purchases.
- perform a more complex solar energy forecasting. This work considers only the power signal generated from the PV and seasonal and day-night periodicity. More complex techniques can be used, bases on weather forecasting and past production history, allowing to have a more complex agent behaviour. The simulator should also be able to operate batteries placed in environments with changing temperatures.
- consider larger battery prices. The current technique only considers discounted batteries, since larger costs causes instabilities in the training process.
- consider unbalanced households, since PV and batteries capacities are selected according to the auto-consumption profile.

Acronyms

DoD Depth of Discharge. 6, 10, 12, 14–16, 20, 21

ESS Energy Storage System. 10, 11

FQI Fitted Q-Iteration. 9, 10, 18, 20, 23

KPI Key Performance Indicator. 18

MDP Markov Decision Process. 7–9
ML Machine Learning. 6

PV Photovoltaic Panel. 2, 12, 15, 17, 23

RL Reinforcement Learning. 7, 9, 11, 23

SEI Solid Electrolyte Interphase. 5, 6
SoC State of Charge. 4–6, 10, 12, 14–16, 19–23
SoH State of Health. 4, 5, 10, 11, 15, 16, 19, 23

References

[1] Xi Fang, Satyajayant Misra, Guoliang Xue, and Dejun Yang. Smart grid—the new and improved power grid: A survey. *IEEE communications surveys & tutorials*, 14(4):944–980, 2011.

[2] Ehsanul Kabir, Pawan Kumar, Sandeep Kumar, Adedeji A Adelodun, and Ki-Hyun Kim. Solar energy: Potential and future prospects. *Renewable and Sustainable Energy Reviews*, 82:894–900, 2018.

[3] Richard S Treptow. Lithium batteries: a practical application of chemical principles. *Journal of chemical education*, 80(9):1015, 2003.

[4] N Kularatna. Rechargeable battery technologies: An electronic engineer's view point. *Energy Storage Devices for Electronic Systems: Rechargeable Batteries and Supercapacitors, Elsevier*, pages 29–61, 2014.

[5] Yujie Wang, Jiaqiang Tian, Zhendong Sun, Li Wang, Ruilong Xu, Mince Li, and Zonghai Chen. A comprehensive review of battery modeling and state estimation approaches for advanced battery management systems. *Renewable and Sustainable Energy Reviews*, 131:110015, 2020.

[6] Marc Doyle, Thomas F Fuller, and John Newman. Modeling of galvanostatic charge and discharge of the lithium/polymer/insertion cell. *Journal of the Electrochemical society*, 140(6):1526, 1993.

[7] Dong Zhang, Branko N Popov, and Ralph E White. Modeling lithium intercalation of a single spinel particle under potentiodynamic control. *Journal of the Electrochemical Society*, 147(3):831, 2000.

[8] VH Johnson. Battery performance models in advisor. *Journal of power sources*, 110(2):321–329, 2002.

[9] Marco Mussi, Luigi Pellegrino, Marcello Restelli, and Francesco Trovò. A voltage dynamic-based state of charge estimation method for batteries storage systems. *Journal of Energy Storage*, 44:103309, 2021.

[10] Bor Yann Liaw, Ganesan Nagasubramanian, Rudolph G Jungst, and Daniel H Doughty. Modeling of lithium ion cells—a simple equivalent-circuit model approach. *Solid state ionics*, 175(1-4):835–839, 2004.

[11] Shuangqi Li, Hongwen He, and Jianwei Li. Big data driven lithium-ion battery modeling method based on sdae-elm algorithm and data pre-processing technology. *Applied energy*, 242:1259–1273, 2019.

[12] Robert Spotnitz. Simulation of capacity fade in lithium-ion batteries. *Journal of power sources*, 113(1):72–80, 2003.

[13] Bolun Xu, Alexandre Oudalov, Andreas Ulbig, Göran Andersson, and Daniel S Kirschen. Modeling of lithium-ion battery degradation for cell life assessment. *IEEE Transactions on Smart Grid*, 9(2):1131–1140, 2016.

[14] Wikimedia Commons. File:rainflow counting example.svg — wikimedia commons, the free media repository, 2020. [Online; accessed 31-March-2022].

[15] Masanori Matsuishi and Tatsuo Endo. Fatigue of metals subjected to varying stress. *Japan Society of Mechanical Engineers, Fukuoka, Japan*, 68(2):37–40, 1968.

[16] Yung-Li Lee and Tana Tjhung. Rainflow cycle counting techniques. *Metal Fatigue Analysis Handbook: Practical Problem-solving Techniques for Computer-aided Engineering*, page 89, 2011.

[17] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[18] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[19] Conor F Hayes, Roxana Rădulescu, Eugenio Bargiacchi, Johan Källström, Matthew Macfarlane, Mathieu Reymond, Timothy Verstraeten, Luisa M Zintgraf, Richard Dazeley, Fredrik Heintz, et al. A practical guide to multi-objective reinforcement learning and planning. *arXiv preprint arXiv:2103.09568*, 2021.

[20] Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6:503–556, 2005.

[21] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[22] Yu Sui and Shiming Song. A multi-agent reinforcement learning framework for lithium-ion battery scheduling problems. *Energies*, 13(8):1982, 2020.

[23] Shiying Huang, Peng Li, Ming Yang, Yuan Gao, Jiangyang Yun, and Changhang Zhang. A control strategy based on deep reinforcement learning under the combined wind-solar storage system. *IEEE Transactions on Industry Applications*, 57(6):6547–6558, 2021.

[24] Amarildo Likmeta, Alberto Maria Metelli, Andrea Tirinzoni, Riccardo Giol, Marcello Restelli, and Danilo Romano. Combining reinforcement learning with rule-based controllers for transparent and general decision-making in autonomous driving. *Robotics and Autonomous Systems*, 131:103568, 2020.

[25] Amarildo Likmeta, Alberto Maria Metelli, Giorgia Ramponi, Andrea Tirinzoni, Matteo Giuliani, and Marcello Restelli. Dealing with multiple experts and non-stationarity in inverse reinforcement learning: an application to real-life problems. *Machine Learning*, 110(9):2541–2576, 2021.

[26] Giorgia Ramponi, Amarildo Likmeta, Alberto Maria Metelli, Andrea Tirinzoni, and Marcello Restelli. Truly batch model-free inverse reinforcement learning about multiple intentions. In *International Conference on Artificial Intelligence and Statistics*, pages 2359–2369. PMLR, 2020.

[27] Jun Cao, Dan Harrold, Zhong Fan, Thomas Morstyn, David Healey, and Kang Li. Deep reinforcement learning-based energy storage arbitrage with accurate lithium-ion battery degradation model. *IEEE Transactions on Smart Grid*, 11(5):4513–4521, 2020.

[28] Alexander JM Kell, A Stephen McGough, and Matthew Forshaw. Optimizing a domestic battery and solar photovoltaic system with deep reinforcement learning. *arXiv preprint arXiv:2109.05024*, 2021.

[29] Anthony Adams and Peter Vamplew. Encoding and decoding cyclic data. *The South Pacific Journal of Natural Science*, 16:54–58, 1998.

[30] Richard L Burden and J Douglas Faires. 2.1 the bisection algorithm. *Numerical analysis*, pages 46–52, 1985.

[31] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[32] Jiuchun Jiang, Wei Shi, Jianming Zheng, Pengjian Zuo, Jie Xiao, Xilin Chen, Wu Xu, and Ji-Guang Zhang. Optimized operating range for large-format lifepo4/graphite batteries. *Journal of The Electrochemical Society*, 161(3):A336, 2013.

Abstract in lingua italiana

Le Smart Grid sono l'evoluzione delle reti elettriche tradizionali e permettono un flusso bidirezionale di elettricità e informazioni tra diversi attori. Ai nodi più periferici di questa rete, i consumatori sono in grado di produrre energia con pannelli fotovoltaici e soddisfare i propri bisogni energetici. Tuttavia, a causa della natura intermittente della produzione di energia solare, queste unità sono caratterizzate da periodi di surplus o deficit di energia. Per risolvere questo problema, è ormai sempre più comune che vengano installati dei set di batterie al Litio che sono usate per conservare l'energia in eccesso per un uso futuro e ridurre scambi energetici costosi con la rete elettrica. Nonostante ciò, questi sistemi di accumulazione sono caratterizzati da un processo di degradazione che ne riduce la capacità massima col passare del tempo.
In questo lavoro di tesi, è stato sviluppato un controllore, basato su tecniche di Reinforcement Learning, per trovare una politica di consumo che massimizzi il profitto economico, tenendo conto sia del processo di degradazione della batteria, sia dei ricavi ottenuti con la compravendita dell'energia stessa. Tale approccio ha portato a un aumento del 15% dei profitti rispetto alle techniche presenti nello stato dell'arte.

Parole chiave: Reinforcement Learning, Smart-Grids, Batterie Al Litio, Fitted Q-Iteration, Controllo, Stato di Salute

Ringraziamenti

Ringrazio dal profondo del cuore Chiara per avermi sostenuto a qualsiasi ora del giorno, Samuele che è stato costretto ad ascoltarmi parlare per ore, il king Tommaso che mi ha aiutato e consigliato innumerevoli sere. Ringrazio Muff e Fabio per la loro pazienza e per l'ospitalità, Matteo con cui ho condiviso questo percorso, Luca ed Erica per tutti i pranzi passati insieme, Enka e Meri per aver trovato tempo nonostante i mille impegni, Morando e Veronica che si sono fatti 30 minuti di macchina per venirmi a trovare, Hakim per le giornate passate insieme in sessione e a sentirlo suonare, Vittorio per avermi sempre tirato su di morale. Ringrazio Marzia, Martina, Willy, Michele e Ferre per la pazienza nonostante tutti i pacchi che ho tirato loro e il poco preavviso nei piani, per aver sopportato le mie lamentele e per avermi sempre sostenuto.