
Shielded Controller Units for RL with Operational Constraints Applied to Remote Microgrids

Hadi Nekoei^{1,2,3}, Alexandre Blondin-Massé⁴, Rachid Hassani⁴, Sarath Chandar^{1,3,5,6}, Vincent Mai⁴

nekoeihe@mila.quebec

¹ Chandar Research Lab ² Université de Montréal ³ Mila – Quebec AI Institute ⁴ IREQ – Hydro-Québec Research Institute ⁵ Polytechnique Montréal ⁶ CIFAR AI Chair

Abstract

Reinforcement learning (RL) is a powerful framework for optimizing decision-making in complex systems under uncertainty—an essential challenge in real-world settings, particularly in the context of the energy transition. A representative example is remote microgrids that supply power to communities disconnected from the main grid. Enabling the energy transition in such systems requires coordinated control of renewable sources like wind turbines, alongside fuel generators and batteries, to meet demand while minimizing fuel consumption and battery degradation under exogenous and intermittent load and wind conditions. These systems must often conform to extensive regulations and complex operational constraints. To ensure that RL agents respect these constraints, it is crucial to provide interpretable guarantees. In this paper, we introduce Shielded Controller Units (SCUs), a systematic and interpretable approach that leverages prior knowledge of system dynamics to ensure constraint satisfaction. Our shield synthesis methodology, designed for real-world deployment, decomposes the environment into a hierarchical structure where each SCU explicitly manages a subset of constraints. We demonstrate the effectiveness of SCUs on a remote microgrid optimization task with strict operational requirements. The RL agent, equipped with SCUs, achieves a 24% reduction in fuel consumption without increasing battery degradation, outperforming other baselines while satisfying all constraints. We hope SCUs contribute to the safe application of RL to the many decision-making challenges linked to the energy transition. All code and supporting data are available at https://github.com/chandar-lab/SCU_RL_MicroGrid.

1 Introduction

Ensuring safe, sustainable, and reliable operation is critical for the control of real-world systems, especially in the context of the energy transition. These systems, such as remote microgrids serving communities without access to the main power grid, must adhere to numerous norms and regulations. Any algorithm deployed on such a system must have its compliance to these constraints explicitly and interpretably proven. The challenge to implement robust and provable compliance guarantees for reinforcement learning (RL) agents hinders their widespread use in critical industrial settings, despite RL’s potential for solving complex optimization problems in real-world applications (Dulac-Arnold et al., 2019; Kober et al., 2013). Addressing this issue is essential for unlocking RL’s full potential in industrial settings, where a wrong decision could lead to equipment damage, system failures, safety incidents, or significant economic losses (García & Fernández, 2015; Amodei et al., 2016).

White-box shielding is a promising approach to meet these requirements (Alshiekh et al., 2018; Jansen et al., 2018). Synthesized from prior knowledge of environment dynamics, a white-box shield verifies that the agent’s action do not violate constraints, and imposes an alternative complying action when necessary. This mechanism, produced in human-understandable logic, can be audited by system experts, and adjusted to the desired level of conservativeness to handle the system’s uncertainties. Significant progress has been made in shield synthesis (Könighofer et al., 2020; Bloem et al., 2020) and white box shielding (Hsu et al., 2023; Krasowski et al., 2023). However, in complex systems with many constraints across multiple devices and varying time horizons, developing a provable shield in a systematic way remains challenging. Indeed, shielding approaches usually require expressing constraints logically, simulating environment rollouts, and designing reasonably performing fallback policies. Doing so for complex environments can be tedious and error prone. A methodology decomposing the shielding problem into simpler sub-problems and allowing compliance to be proved independently for each constraint would considerably facilitate this process, enhancing the applicability of RL in critical settings.

In this paper, we propose Shielded Controllers Units (SCUs) for RL, a systematic methodology aimed at this objective. The environment is decomposed into SCUs, which link a controller to a real system to ensure the compliance of a corresponding set of constraints. Systems can be devices or a group of lower-level SCUs, forming a hierarchical structure. A controllers consists of a shielded action dispatcher and a digital twin of its corresponding system. The digital twin combines a state estimator, updated from sensor measurements, and a simulator to predict future system behavior. The shielded action dispatcher uses prior knowledge and the digital twin to test the action’s validity before relaying it to the real system. Each SCU possesses a shield to guarantee compliance with its associated set of constraints. The SCU structure ensures full compliance over the whole system, while considerably simplifying the shield design process for complex real-world environment.

We demonstrate SCUs on the problem of remote microgrid optimization. Remote microgrids are small power systems disconnected from the main grid, typically powered by fossil fuel generators (*gensets*). To reduce CO2 emissions and operational costs, electricity utilities are installing wind turbines for fossil-free energy. Due to the intermittent nature of wind power, large capacity batteries are needed for flexibility, while gensets remain essential to reliably meet power demand. The power dispatch problem aims to optimize the use of batteries and gensets to minimize fuel consumption and battery degradation. This complex sequential decision-making problem involves long-term planning, uncertain exogenous variables such as power demand and wind, and many constraints to ensure reliable and sustainable operation. When modeled in a realistic industrial scenario, it is a difficult problem for non-learning based optimization methods, making it an excellent use case to demonstrate the power of the shielded controllers: our RL agent outperforms baselines while its compliance to every constraint is guaranteed. This paper makes three contributions:

- It introduces Shielded Controller Units (SCUs), a systematic approach to white-box shield design, improving interpretability while ensuring constraint compliance for RL agents in industrial contexts.
- It presents a realistic remote microgrid optimization problem under uncertainty, aimed at training and testing an RL agent with real-world operational constraints.
- The SCU approach is applied to microgrid optimization, ensuring constraint compliance while the resulting RL agent outperforms current baselines for this problem.

2 Related work

Constrained RL seeks to optimize policies under safety constraints and is an active area of research (Wachi et al., 2024; Liu et al., 2021). Black-box methods like CPO (Achiam et al., 2017) and CVPO (Liu et al., 2022) rely on cost signals to discourage unsafe behavior but often lack interpretability and struggle with learning (Mani et al., 2025). Shielding approaches that train classifiers to block unsafe actions (Waga et al., 2022) face similar issues and depend heavily on environmen-

tal knowledge. White-box shielding instead leverages known dynamics. Early work (Bloem et al., 2012; Alshiekh et al., 2018) used formal methods to synthesize shields, while later efforts incorporated runtime monitoring (Fulton & Platzer, 2018), probabilistic safety constraints (Bouton et al., 2019), and adaptive interventions via model predictive control (Banerjee et al., 2024). Reviews (Krawowski et al., 2023; Hsu et al., 2023) summarize the tradeoffs between black-box and model-based approaches. However, these approaches can be difficult to deploy in practice. They often require formal logic, simulated rollouts, and/or fallback policies, which are tedious and error-prone to design in complex systems. We address this by introducing a practical, modular methodology that decomposes safety enforcement into simpler shielding sub-problems, making shield design easier. The closest to our work is factored shielding (ElSayed-Aly et al. 2021), which improves scalability in multi-agent RL by assigning localized shields to subsets of agents equivalent to horizontal decomposition across agents. Our SCU-based approach instead leverages hierarchical decomposition of the environment and digital twins for modular, interpretable shielding for complex real-world systems.

Microgrid optimization aims to manage energy sources for reliable, low-emission, and sustainable operation, accounting for battery degradation. This requires long-term planning under uncertainty (e.g., power demand, wind), which challenges traditional methods due to mixed variables and non-linear dynamics. Mixed-integer programming has had some success (Hajimiragha & Zadeh, 2013; Silvente et al., 2015; Hirwa et al., 2022), but often oversimplifies constraints and ignores battery depletion. A more realistic model (Lambert & Hassani, 2023) supports real-time control of gensets but excludes batteries and wind.

RL-based approaches show promise (Dimeas & Hatziaegyriou, 2010; Foruzan et al., 2018; Arwa & Folly, 2020; Yang et al., 2021; Wang et al., 2024), though often lack industrial realism or safety guarantees. Eichelbeck et al. (2022) apply RL for energy dispatch under similar constraints, using an action projection shield, though replacing fuel genset dynamics with islanding robustness. Ceusters et al. (2023) train an RL agent with a physics-based safety layer and fallback policy for a multi-energy system. Other constrained RL work spans related domains: cooling (Yu et al., 2024), EV charging (Zhang et al., 2024), and voltage regulation (Chen et al., 2021).

3 Shielded controller units

In complex, critical environments like real-world industrial problems, numerous constraints must be respected across systems with multiple devices. Some constraints pertain to individual devices, while other concern groups of devices, binding their control together. Ensuring compliance with a constraint sometimes requires considering future environment states to guarantee that an action is recoverable. To satisfy operational norms, any agent deployed on such complex system must inter-pretably ensure compliance with these constraints, despite uncertainties from exogenous variables.

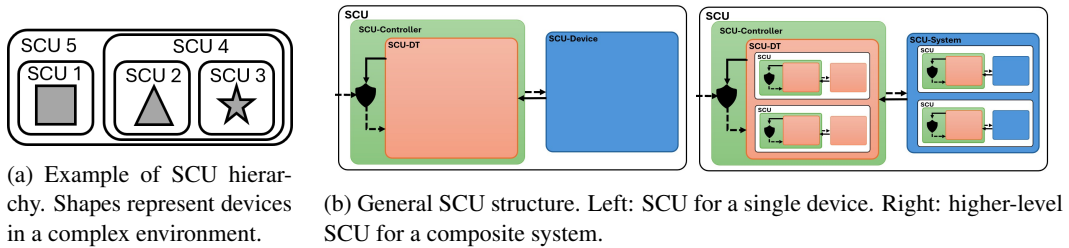


Figure 1: Illustrations of Shielded Controller Units (SCUs) at different levels of abstraction. Our proposed methodology addresses this problem systematically by decomposing the environment into Shielded Controller Units (SCUs). This decomposition is based on analyzing the set of constraints. Every constraint pertains to a subset of the environment’s devices, and constraints corresponding to the same subset are implemented in the same SCU. SCUs are implemented hierarchically, meaning they cannot overlap without one containing the other. For example, consider a

system composed of square, triangle and star devices as shown in Figure 1a. Each device has its own constraints, and thus its own SCU: 1, 2 and 3. Two additional constraints might concern the triangle and the star: they would be managed by SCU 4. If another constraint links the triangle and square devices, whether or not the star is involved, it cannot be treated outside of SCU 4. Therefore, SCU 5 will contain both SCU 1 and SCU 4.

3.1 Elements of shielded controller units

An SCU links a shielded controller to a real system. The controller receives an action from a higher-level SCU, ensures it respects constraints at its level, and relays it to the real system. It contains two elements: a digital twin and a shielded dispatcher. The shielded dispatcher enforces constraints by checking each action and sending a corrected one if needed for every system subcomponent. Many white-box shielding methods exist; choices depend on constraint type and prior knowledge. Some rely on state estimation (Carr et al., 2022) or model predictive shielding (Bastani, 2020; Banerjee et al., 2024). This need is met by the digital twin, combining a state estimator updating internal state from sensors and a simulator predicting system behavior from the current state and action sequence. Importantly, the digital twin must simulate all SCUs of components to ensure compliance with constraints. This includes the digital twin for the real system *and* the digital twin inside the SCU controller, as shown in Figure 1b.

3.2 Step function and interactions

The SCU structure operates hierarchically, processing the action sent by the agent from the top level SCU down to the lower levels, and propagating the information from the devices sensors back up to the agent while updating the various levels of digital twins. This process can be realized with a single step function called from the top SCU, as described in algorithm 1. It is divided in three substeps.

1. First, at line 3, the shield dispatcher receives action a_t from the higher level and dispatches it into complying actions $\{a_t^{\text{comp},i}\}_{i=1\dots k}$ using function SHIELD. This function is designed ad-hoc via an interpretable shielding method, potentially with digital twin U^{DT} .
2. Next, lines 4 to 11, the safe actions are propagated to the real system's STEP function. If it is a device, it returns observation o_{t+1} . If it is a system of SCUs, it will return set $\{\hat{s}_{t+1}^i\}_{i=1\dots k}$ of state estimations of the subcomponents' SCUs. These are aggregated in observation o_{t+1} .
3. Finally, at line 12, the controller internal state s_{t+1}^{SC} and its numerical twin's $s_{t+1}^{\text{SC:DT}}$ are updated using function UPDATECONTROLLER in algorithm 2. s_{t+1}^{SC} is updated with ad-hoc function CONTROLLERSTATE and $s_{t+1}^{\text{SC:DT}}$ with function UPDATEDT. If the SCU is of a device, UPDATEDT estimates s_{t+1}^{DT} using ad-hoc estimation function STATEESTIM. If however the SCU has k SCUs as sub-components U^j , there are 2 elements to update per U^j : (1) digital twin $U^{j,\text{DT}}$ of the real device and (2) controller $U^{j,\text{SC}}$ and its internal digital twin $U^{j,\text{SC:DT}}$. This implies a new recursive function propagating o_{t+1} to lower levels j . It is disaggregated into o_{t+1}^j which are used for UPDATEDT on $U^{j,\text{DT}}$ and UPDATECONTROLLER on $U^{j,\text{SC}}$. Finally, the digital twin new state s_{t+1}^{DT} is the aggregation of lower level SCU states s_{t+1}^j .

Algorithm 2 Controller update function

```

1: function UPDATECONTROLLER( $U^{\text{SC}}, o_{t+1}$ )
2:    $s_{t+1}^{\text{SC}} \leftarrow \text{ControllerState}(U^{\text{SC}}, o_{t+1})$ 
3:    $s_{t+1}^{\text{SC:DT}} \leftarrow \text{UpdateDT}(U^{\text{SC:DT}}, o_{t+1})$ 
4:    $s_{t+1} = [s_{t+1}^{\text{SC}}, s_{t+1}^{\text{SC:DT}}]$ 
5:   return  $s_{t+1}$ 
6: end function

```

Algorithm 1 SCU step function

```
1: function STEP( $U, a_t$ )  $U$ : SCU,  $a_t$ : action
2:    $k \leftarrow U.\text{NbCOMPONENTS}$ 
3:    $\{a_t^{\text{comp},i}\}_{i=1\dots k} \leftarrow \text{SHIELD}(U^{\text{DT}}, a_t)$ 
4:   if  $U.\text{ISDEVICE}$  then  $k = 1$ 
5:      $o_{t+1} \leftarrow \text{STEP}(U^{\text{dev}}, a_t^{\text{comp},1})$ 
6:   else  $k > 1$ 
7:     for  $i \in \{1 \dots k\}$  do
8:        $s_{t+1}^i \leftarrow \text{STEP}(U^i, a_t^{\text{comp},i})$ 
9:     end for
10:     $o_{t+1} \leftarrow \text{OBS}(\text{AGG}(\{s_{t+1}^i\}_{i=1\dots k}))$ 
11:   end if
12:    $s_{t+1} \leftarrow \text{UPDATECONTROLLER}(U^{\text{SC}}, o_{t+1})$ 
13:   return  $s_{t+1}^{\text{DT}}$ 
14: end function
```

Algorithm 3 Digital twin update function

```
1: function UPDATEDT( $U^{\text{DT}}, o_{t+1}$ )
2:   if  $U.\text{ISDEVICE}$  then
3:      $s_{t+1}^{\text{DT}} \leftarrow \text{StateEstim}(s_t^{\text{DT}}, o_{t+1})$ 
4:   else
5:      $k \leftarrow U.\text{NbCOMPONENTS}$ 
6:      $\{o_{t+1}^j\}_{j=1\dots k} \leftarrow \text{Disagg}(o_{t+1})$ 
7:     for  $j \in 1 \dots k$  do
8:        $s_{t+1}^{j,\text{SC-DT}} \leftarrow \text{UpdateDT}(U^{j,\text{DT}}, o_{t+1}^j)$ 
9:        $s_{t+1}^j \leftarrow \text{UpdateController}(U^{j,\text{SC}}, o_{t+1}^j)$ 
10:    end for
11:     $s_{t+1}^{\text{DT}} \leftarrow \text{Agg}(s_{t+1}^j)$ 
12:   end if
13:   return  $s_{t+1}^{\text{DT}}$ 
14: end function
```

3.3 Integration within the RL framework

The RL agent interacts with the environment using the usual Markov Decision Process (MDP) framework, but only through the highest level SCUs' step functions. These receive their respective parts of the action space's inputs and propagate them to their subcomponents down to the devices. The agent's observation is the aggregation of high-level SCUs' controllers' state estimation.

4 Shielded remote microgrid optimization

We consider a remote microgrid composed of a wind turbine, a battery and two fuel gensets, generating electric power to meet the demand of a village. The objective of the power dispatch problem is to control these four devices to minimize genset fuel consumption and battery depreciation, while ensuring that the generated power satisfies the demand at every time step. In this problem, two variables are considered exogenous and beyond the agent's control: demand and available wind power. Specialized industry models can provide predictions to the agent, but they are uncertain. This problem, when realistically modeled to target industrial deployment, is an excellent test case for the SCU approach. It is of reasonable scale for in-depth analysis but complex enough that non-learning based methods struggle with real time control. RL is a powerful alternative, but it must provably respect the industrial constraints. In this section, we describe how we modeled the environment's physical dynamics, rewards and constraints. We also discuss how the environment is decomposed in SCUs (See Figure 2), the shielding approach for each controller, and define the MDP the RL agent actually interacts with.

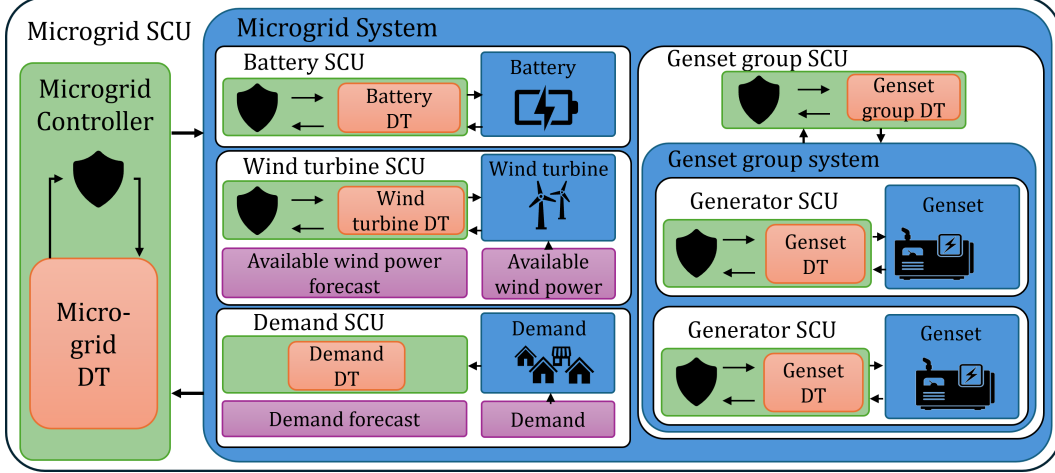


Figure 2: SCU decomposition of the microgrid.

4.1 Exogenous variables

There are two exogenous variables in the environment, over which the agent has no control: the power demand δ_t and the available wind power $p_t^{\text{wind,avail}}$. We use real data gathered from remote microgrids over one year. Demand ranges from 180 to 540 kW, averaging 320 kW, while available wind power ranges from 0 to 400 kW, averaging 272 kW. The agent observes both variables at every time step and receives realistic predictions over a 30-point forecast at 15 minutes intervals, covering 7.5 hours. These predictions were generated using industry-standard approaches. Demand forecasts were produced using a general additive model (GAM) that combines transformations of the consumption observed the previous day and the previous week, along with observed and forecast temperatures (David et al., 2023). Wind power forecasts were generated by downscaling meteorological variables and converting them using an 8-direction power curve model. Forecasts were then fine-tuned using turbine-specific loss models, availability, and a statistical auto-adaptive model with real-time data.

4.2 Devices

The modeled microgrid includes three types of device: one wind turbine, two gensets, and one battery. Each device is managed by an SCU. The industrial norms they must comply with are described in **bold**. In this experiment, the digital twin sensor update function for devices and systems has been implemented as exact state update, which is realistic given the availability of precise sensors to measure relevant state variables for batteries, gensets and wind turbines.

Wind turbine The wind turbine generates renewable electric power for the microgrid by harnessing wind energy at no cost. At step t , it receives action $a_t = \bar{p}_t^{\text{wind}}$ containing the wind power curtailment setpoint. The resulting power p_t^{wind} is $\bar{p}_t^{\text{wind}}$ clipped between 0 and current available wind power $p_t^{\text{wind,avail}}$. In this power dispatch problem, **no operational constraints applies** on the wind turbine, so the shield directly relays $a_t^{\text{comp}} = a_t$ to the device.

Battery The battery stores chemical energy and can absorb or release electrical power p_t^{batt} . The power p_t^{batt} responds to the $\bar{p}_t^{\text{batt}}$ setpoint from action a_t , can be positive (discharge) or negative (charge), and is limited by the battery nominal power of 600 kW. The battery state of charge (SoC) indicates the accumulated energy as a percentage of the battery 672 kWh capacity. Charging and discharging are inefficient processes, leading to a 5% energy loss in each direction. We model the battery’s physical behavior precisely according to Hassani & Lambert (2024). The battery’s degradation over charging/discharging cycles $d_{b,t}$ is modeled in arbitrary units using an approximated online rainflow algorithm adapted for MDPs, inspired from Obermayr et al. (2021) and Kwon &

Zhu (2022) and detailed in the appendix. To prevent damage to the battery, the **SoC is constrained between 90 and 10% of capacity**, or 5% as a reserve in case of emergency. The shield uses prior knowledge of the battery inner dynamics to determine the corresponding limits and accordingly clips setpoint $\bar{p}_t^{\text{batt}}$ received from a_t before relaying it as a_t^{comp} to the real battery.

Gensets Gensets can produce electric power from fuel energy. We model their behavior and constraints based on Lambert & Hassani (2023). Gensets receive action $a_t = (\Delta_t^{\text{gen}}, \bar{p}_t^{\text{gen}})$. Gensets's running status can be *on* or *off*, changed by *start* or *stop* command Δ_t^{gen} . Once *on*, their power production p_t^{gen} is controlled by setpoint $\bar{p}_t^{\text{gen}}$ ranging between 0 and maximum power of 440 kW. Fuel consumption is proportional to p_t^{gen} at 0.25 l/kWh, plus a constant 10 l/h when the genset is *on*.

To prevent long term damage, several operational constraints must be respected. When changing status, **gensets must complete routines**, like a 3-minute warm-up producing 100 kW or a 5-minute cool-down producing 0 kW. Once *on*, they have **minimum runtime** of 30 minutes during which they should not be turned *off*. Except during these routines, *on* gensets have a **minimal power production** of 120 kW. Producing the maximum 440 kW continuously can damage the device: **it should instead be capped at its nominal capacity** of 400 kW, except in emergencies where the 40 kW overload reserve can be used. Finally, to meet ISO 8528-1 standards (International Organization for Standardization (ISO), 2018), the **average power over the last 48 hours of operation is capped** at 70% of nominal capacity. The genset SCU controller ensures these constraints are respected by modifying violating Δ_t^{gen} and clipping $\bar{p}_t^{\text{gen}}$ from a_t before relaying them as a_t^{comp} to the physical genset.

4.3 Systems

Two sets of operational constraints for the microgrid concern specific groups of devices, leading to two additional SCUs: the genset orchestrator and the microgrid.

Genset orchestrator The genset orchestrator manages the two gensets, its subcomponents, by dispatching action $a_t = (\Delta_t^{\text{orch}}, \bar{p}_t^{\text{orch}})$ into actions $a_t^{\text{comp},i} = (\Delta_t^{\text{gen},i}, \bar{p}_t^{\text{gen},i})$ complying with two operational constraints. First, genset i cannot be turned *on* if genset $i-1$ is turned *off*, and vice versa. This **priority order constraint** ensures compatibility with existing heuristic-based systems so they can take over if necessary, as described in Hassani & Lambert (2024). The shielded dispatcher follows a state-machine logic allowing it to turn *on* or *off* only one specific generator at the time. This allows to dispatch a_t 's single status change command Δ_t^{orch} to *start* or *stop* a generator, or keep the current configuration, which the shield defaults to if Δ_t^{orch} is inapplicable.

The other constraint is the **equal power fraction** presented in Lambert & Hassani (2023): all running genset not in warmup or cooldown must run at the same percentage of their nominal powers. The orchestrator uses its digital twin to account for single genset constraints, such as the 70% maximum average power over 48h, as it may limit both gensets together. This constraint binds all $\bar{p}_t^{\text{gen},i}$ together: the orchestrator dispatches them in $a_t^{\text{comp},i}$ so that the total power produced by the gensets, p_t^{orch} , is as close as possible to the orchestrator setpoint $\bar{p}_t^{\text{orch}}$ received in a_t .

Microgrid The microgrid is the highest level SCU. Its objective is to ensure that **the balance, i.e. the difference between power generation and demand, is kept at 0 for all time steps**. A negative balance, or shortage, is a critical failure of the system, while a positive balance should also be avoided although it is easier to recover from. To enforce this constraint, the microgrid subcomponents are the battery SCU, the wind turbine SCU, and the genset orchestrator SCU. Each time step, the microgrid shield dispatches complying actions $a_t^{\text{comp},i}$ to its three subcomponents, as power setpoints $\bar{p}_t^{\text{batt}}$, $\bar{p}_t^{\text{wind}}$ and $\bar{p}_t^{\text{orch}}$, and orchestrator status change Δ_t^{orch} . The balance constraint, and a hard-coded wind over fuel priority, reduce the degrees of liberty of the received action a_t to Δ_t^{orch} and $\bar{p}_t^{\text{batt}}$.

To ensure 0 balance, shielding operates at two levels. At any given time step t , the subcomponents setpoints must ensure the sum of generated powers p_t^{batt} , p_t^{wind} and p_t^{orch} meets demand δ_t . It

priorizes respecting the agent’s battery setpoint, then using wind before fuel. If the demand is not met due to the limits of wind and gensets, $\bar{p}_t^{\text{batt}}$ is adjusted. If all the wind power cannot be absorbed by the demand and the battery, it is curtailed. To ensure that every subcomponent constraint is accounted for, the microgrid digital twin simulates each SCU’s response to given commands.

The shield must also avoid actions leading to irrecoverable future states, characterized by the genset orchestrator’s inability to provide the power to balance the demand due to a genset being in cool-down/warm-up routine. Such blockages can take a maximum of 8 minutes. To prevent this, a *recovery shield* using a predictive shielding approach is deployed (Bastani, 2020; Hsu et al., 2023). Monitoring is done by simulating the environment with the digital twin over a 9-minute period, given a *fallback* genset orchestrator policy $\{\Delta_t^{\text{orch}}, \Delta_{t+1}^{\text{orch}}, \dots, \Delta_{t+8}^{\text{orch}}\}$ with future $\Delta_{t+\tau}^{\text{orch}} = \text{Start}$ for $\tau \in 1, \dots, 8$. There are three unknown variables when simulating the next steps: exogenous demand, available wind power, and agent battery commands. The recovery shield guarantee should hold irrespective of the precision of exogenous variable predictions, so they are not used here. Instead, two scenarios are considered: (1) The worst case scenario with reserve, where battery command is set to discharge to the maximum, and demand and wind are simulated as rising and decreasing, respectively, towards their historical high and low at their fastest historical rate of change. The genset and battery devices are simulated as able to access their emergency reserve power. (2) A constant scenario for wind and demand, but where simulated gensets and batteries are not allowed to access their reserves. Scenario (1) ensures a strong guarantee on the 0-balance constraint, while the (2) reduces reliance on reserve. If a negative balance is detected during one of these simulations, Δ_t^{orch} is considered as possibly non-compliant and is replaced by the least different recoverable option.

4.4 Markov Decision Process for the agent

As the microgrid SCU is at the highest level, it interfaces with the RL agent. The agent’s action space is two-dimensional: a discrete value for Δ_t^{orch} and a continuous value for $\bar{p}_t^{\text{batt}}$. Its observation space includes the state estimation of the microgrid subcomponents and two time series of exogenous variable predictions – wind and demand. The reward is a negative cost combining the gensets fuel consumption $\mathbf{F}_{o,t}$ (in liters) and the battery degradation $d_{b,t}$ in arbitrary units: $r_t = -(\mathbf{F}_{o,t} + \alpha d_{b,t})$, where weight α can be adjusted based on the utility’s objective. Between equally performing solutions in terms of cost, a controller minimizing reserve usage of batteries and gensets is preferred.

5 Experimental results

5.1 Experimental setup

We used the Soft-Actor Critic (SAC) RL algorithm (Haarnoja et al., 2018), which is known for its robust performance in continuous domains, to train our agents. Although we also trained the RL agents using PPO (Schulman et al., 2017), we found that SAC consistently outperformed PPO in this setting. Therefore, we report the results for the SAC agents. To process the available wind power and demand prediction time series, the agent’s architecture integrates LSTMs. Details about the agent’s architecture can be found in the appendix. The agents were trained on 10^5 one-minute environment steps divided in one-day episodes. The episodes initialization involved sampling a random time during the year and the gensets and battery initial states. We explored several hyperparameter configurations (detailed in the appendix) as well as 10 training seeds and three values of α for the SAC agent. Both the baselines and the trained agents were then tested in a fixed test environment consisting of the first 10-day periods of each month from February to November to represent varying conditions. To report the results, we averaged the performance across these 100 days. For the RL agent’s performance, we also averaged the performance and calculated the standard errors across the 10 training seeds.

Note that all agents use shields during evaluation unless stated otherwise. We verified that none of the constraints are violated to confirm the validity of the shield.

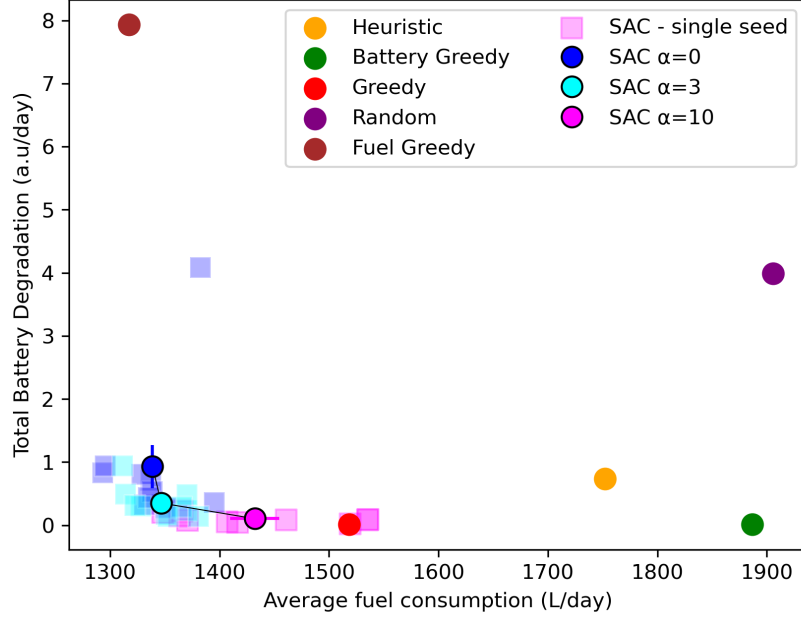


Figure 3: The RL agent seeks to strike a balance between fuel consumption and battery degradation costs. Changing their respective importance in the reward function with α seems to exhibit the Pareto frontier. In contrast, all other baselines exhibit suboptimal performance in one or both metrics.

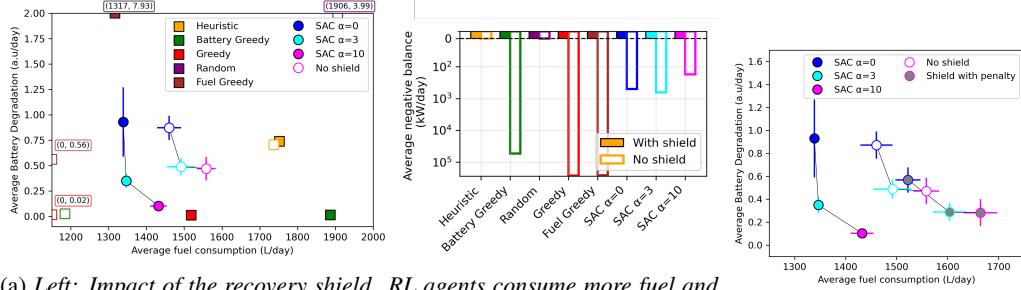
5.2 Results

We designed our experimental section to address three research questions: (1) How do shielded RL agents perform compared to baseline algorithms in terms of minimizing fuel consumption and battery degradation? (2) How does the recovery shield impact the performance and compliance to constraints? (3) Given a shielded deployment environment, how to constrain an RL agent during training?

5.2.1 Performance analysis

Figure 3 compares the performance of our RL agents with four control baselines providing action $a_t = (\Delta_t^{\text{orch}}, \bar{p}_t^{\text{batt}})$ based on predetermined policies. The *random* agent uniformly samples discrete Δ_t^{orch} from (*start*, *stop*, *do nothing*), and continuous $\bar{p}_t^{\text{batt}}$ from maximum discharging to maximum charging power. It performs poorly, showing that relying on the SCU shield is not enough to guarantee good performance. The *battery greedy* agent lets the shield change the genset status and keeps the battery idle, with Δ_t^{orch} to *do nothing*, and $\bar{p}_t^{\text{batt}}$ to 0. The battery does not degrade but fuel consumption is not optimal. The *greedy* agent tries to turn the genset off with Δ_t^{orch} at *stop*, and discharges the battery with maximal $\bar{p}_t^{\text{batt}}$. It consumes little fuel but highly degrades the battery. The *fuel-greedy* agent also tries to turn the genset off but charges the battery when available wind power is higher than the demand, and discharges otherwise.

The industry *heuristics* reproduces a policy currently deployed in industrial settings. It always keeps one genset *on*, and decides to turn the other genset *on* when current genset is at more than 90% of its available power (nominal or capped by the moving average constraint) for 5 minutes in a row, or directly if it reaches more than 100%. It turns the second genset *off* if the current total genset orchestrator power could be managed by 70% of the other genset’s available power for 5 minutes in a row. On the battery side, it switches between two modes: charging, or discharging, to avoid fast cycling. In charging mode, it only charges $\bar{p}_t^{\text{batt}}$ to capture an eventual excess power from the wind.



(a) Left: Impact of the recovery shield. RL agents consume more fuel and degrade the battery more without shielding. Right: shielding prevents balance violations. (b) Shield penalty during training reduces performance.

Figure 4: Performance of agents with and without recovery shielding in remote microgrid environments.

It does not use the genset power to charge. Discharge only happens in case of emergency, as enforced by the shield. The battery charges until achieving 90% SoC, and then switches to discharging mode. In discharging mode, $\bar{p}_t^{\text{batt}}$ is set to maximum discharging power until the battery reaches 10% SoC. It then switches back to charging mode. Performance-wise, it provides a relevant comparison point in terms of trade-off between both metrics.

The RL agents trained with SAC perform significantly better, improving fuel efficiency by 24% for similar battery degradation. Modifying parameter α to balance battery degradation cost and fuel consumption in the reward function seems to exhibit a Pareto frontier. This could allow the electric utilities to trade-off battery lifetime and fuel efficiency at their preference while striking the high gain zone for remote microgrid fuel consumptions.

5.2.2 Impact of the recovery shield

In this experiment, we explore the importance of recovery shielding for both RL agents and baseline algorithms. The recovery shield is the most complex shield in this system, addressing the uncertainty of exogenous variables and requiring multi-step simulation for compliance verification. To ensure compliance in every situation, the recovery shield considers worst-case scenario which could impair performance due to over restrictiveness. Figure 4a presents the performance metrics and negative balance violations, with and without recovery shield. Battery-greedy agent without recovery shield has a relatively good performance but it is not acceptable due to violating the negative balance constraint. Fuel-greedy agent without recovery shield does barely consume any fuel but it also show high amount of negative balance violation. On the other hand, Heuristic policy has a stable performance even without recovery shield which shows the validity of its policy. Finally, SAC agents trained and deployed without recovery shield, but penalized for negative balance violation exhibit increased fuel consumption and slightly higher battery degradation than their shielded counterparts, while still sometimes experiencing negative balance (as shown in the right plot). We hypothesize that this occurs because they adopt an overly conservative and suboptimal policy due to the substantial penalties associated with negative balance during training. It is noteworthy that the RL agent with shielding shows much smaller standard errors in evaluated variables, demonstrating the robustness of training the RL agent with a shield.

5.2.3 How to train an RL agent given a shield?

A recurring research question is how penalizing the use of shielding during training impacts the agent’s ability to learn an optimal policy (Hsu et al., 2023). We explore this question for our use case by evaluating three variants of the RL agents in Figure 4b: the first variant (full color) was trained with recovery shielding but without penalizing the shield intervention. The second variant (gray) was trained with recovery shield and a penalty for shield intervention, and the third agent

(white) was trained without recovery shield but with a penalty for violating the non-zero balance constraint.

The agent trained with recovery shielding and no penalty (blue) achieves best performance. Both the orange and green agents learn to be more conservative due to the negative balance or shield penalty. Over-conservativeness is a known issue in punishing RL environments (Mani et al., 2025). Indeed, we notice these agents keep gensets *on* longer which decreases the chance of receiving a penalty. Interestingly, the shield penalty seems to encourage the agent to charge the battery more probably because the shield penalty is harder to predict than the negative balance penalty for the agent.

6 Discussion

While remote microgrid optimization is complex, it remains at a sufficiently small scale that it can be analyzed in depth. Eichelbeck et al. (2022) have shown that, for a very similar problem, a provable, single shield can be hand-designed to respect constraints without significantly impairing the performance. However, their paper shows the challenge of designing such a shield to comply with all constraints at once. Our approach’s applicability stands out in comparison: for every SCU, the shield is simple as it only focuses on a specific set of constraints: the others are accounted for by the SCU structure. In most cases, fallback policies are evident. Handling uncertain dynamics, such as with exogenous variables for the microgrid SCU, can be managed with the necessary level of restrictiveness. In addition, SCUs allow to analyze the degrees of freedom that each constraints removes from the RL agent, and allow it to focus on these instead of outputting actions for every device.

A potential limitation of the SCU approach is the duplication of lower-level digital twins inside higher-level digital twins. This allows higher-level shielded controllers to consider the lower-level constraints when using rolling out strategies, but could lead to an exponential number of digital twins depending on the number of SCU levels. However, in moderately complex real world cases, the number of SCU levels can be expected to be small (Siyari et al., 2019), such that scaling should not be a limiting factor. As an example, in our case the SCU hierarchy has depth 3, and each microgrid SCU step—including future projections—executes in under 0.05 seconds on a standard CPU, demonstrating that practical deployments remain well within real-time control.

7 Conclusion

This paper presented Shielded Controller Units, a decomposition approach facilitating the design of white-box shields to ensure compliance for complex constrained systems controlled by RL agents. SCUs allow to design simple shields for every constraint, which are then combined hierarchically. Digital twins pertaining to each SCU are updated at every time step so higher-level units shields can rollout simulations accounting for lower-level constraints. SCUs were applied on a realistic remote microgrid optimization problem, ensuring the RL agent complies with every operational constraints. Every shield level was simple to design, describe and interpret, showcasing the relevance of SCU for provable RL compliance in complex real-world settings.

By facilitating deployment of RL to critical systems requiring interpretable guarantees, SCUs can allow us to benefit from the potential of RL for optimizing decision-making with important social impact. We demonstrated this by showing how RL agents when supported by SCUs can reduce fuel consumption by 24% for similar battery degradation compared to industry heuristics. We hope SCUs can be valuable for RL deployment in other regulated systems in the fields of energy, healthcare, transportation, etc.

Broader Impact Statement

SCUs aim at facilitating the proof of constraint compliance for RL agents in complex systems and thus enhance RL applicability to critical industrial settings. This is a necessary step towards capitalizing on the potential of RL for real world applications. In the case of power systems such

as microgrids, this could facilitate the integration of renewable energy sources towards the energy transition.

Acknowledgments

The authors would like to thank Mathieu Lambert for his precious advice, and Slavica Antic for her help gathering wind turbine data.

References

- Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization, 2017. URL <https://arxiv.org/abs/1705.10528>.
- Mohammed Alshiekh, Roderick Bloem, Robert Könighofer, Laura Nimmermann, Anna-Katharina Schmuck, and Stefan Weinberger. Safe reinforcement learning via shielding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.
- Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.
- E. O. Arwa and K. A. Folly. Reinforcement learning techniques for optimal power control in grid-connected microgrids: A comprehensive review. *IEEE Access*, 8:208992–209007, 2020. DOI: 10.1109/ACCESS.2020.3038735.
- Arko Banerjee, Kia Rahmani, Joydeep Biswas, and Isil Dillig. Dynamic model predictive shielding for provably safe reinforcement learning. (arXiv:2405.13863), Dec 2024. URL <http://arxiv.org/abs/2405.13863>. arXiv:2405.13863 [cs].
- Osbert Bastani. Safe reinforcement learning with nonlinear dynamics via model predictive shielding, 2020. URL <https://arxiv.org/abs/1905.10691>.
- Roderick Bloem, Barbara Jobstmann, Nir Piterman, Amir Pnueli, and Yaniv Sa’ar. Synthesis of reactive (1) designs. *Journal of Computer and System Sciences*, 78(3):911–938, 2012.
- Roderick Bloem, Peter Gjørl Jensen, Bettina Könighofer, Kim Guldstrand Larsen, Florian Lorber, and Alexander Palmisano. It’s time to play safe: Shield synthesis for timed systems. (arXiv:2006.16688), Jun 2020. DOI: 10.48550/arXiv.2006.16688. URL <http://arxiv.org/abs/2006.16688>. arXiv:2006.16688 [cs].
- Maxime Bouton, Jesper Karlsson, Alireza Nakhaei, Kikuo Fujimura, Mykel J. Kochenderfer, and Jana Tumova. Reinforcement learning with probabilistic guarantees for autonomous driving. (arXiv:1904.07189), May 2019. DOI: 10.48550/arXiv.1904.07189. URL <http://arxiv.org/abs/1904.07189>. arXiv:1904.07189 [cs].
- 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, September 2020. ISSN 1949-3053, 1949-3061. DOI: 10.1109/TSG.2020.2986333.
- Steven Carr, Nils Jansen, Sebastian Junges, and Ufuk Topcu. Safe reinforcement learning via shielding under partial observability. (arXiv:2204.00755), Aug 2022. DOI: 10.48550/arXiv.2204.00755. URL <http://arxiv.org/abs/2204.00755>. arXiv:2204.00755 [cs].
- Glenn Ceusters, Luis Ramirez Camargo, Rüdiger Franke, Ann Nowé, and Maarten Messagie. Safe reinforcement learning for multi-energy management systems with known constraint functions. *Energy and AI*, 12:100227, Apr 2023. ISSN 2666-5468. DOI: 10.1016/j.egyai.2022.100227.

-
- Yize Chen, Yuanyuan Shi, Daniel Arnold, and Sean Peisert. Saver: Safe learning-based controller for real-time voltage regulation. (arXiv:2111.15152), Nov 2021. DOI: 10.48550/arXiv.2111.15152. URL <http://arxiv.org/abs/2111.15152>. arXiv:2111.15152 [cs, eess].
- Charline David, Alexandre Blondin Massé, and Arnaud Zinflou. Fast short-term electrical load forecasting under high meteorological variability with a multiple equation time series approach. *International Journal of Electrical and Computer Engineering*, 17(10):234 – 241, 2023. ISSN eISSN: 1307-6892. URL <https://publications.waset.org/vol/202>.
- A. L. Dimeas and N. D. Hatziaargyriou. Multi-agent reinforcement learning for microgrids. In *IEEE PES General Meeting*, pp. 1–8, July 2010. DOI: 10.1109/PES.2010.5589633.
- Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. Challenges of real-world reinforcement learning. *arXiv preprint arXiv:1904.12901*, 2019.
- Michael Eichelbeck, Hannah Markgraf, and Matthias Althoff. Contingency-constrained economic dispatch with safe reinforcement learning. In *2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 597–602, Dec 2022. DOI: 10.1109/ICMLA55696.2022.00103. URL <http://arxiv.org/abs/2205.06212>. arXiv:2205.06212 [cs, eess].
- E. Foruzan, L.-K. Soh, and S. Asgarpour. Reinforcement learning approach for optimal distributed energy management in a microgrid. *IEEE Transactions on Power Systems*, 33(5):5749–5758, September 2018. DOI: 10.1109/TPWRS.2018.2823641.
- Nathan Fulton and André Platzer. Safe reinforcement learning via formal methods: Toward safe control through proof and learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(11), Apr 2018. ISSN 2374-3468. DOI: 10.1609/aaai.v32i1.12107. URL <https://ojs.aaai.org/index.php/AAAI/article/view/12107>.
- Javier García and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.
- Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870. PMLR, 10–15 Jul 2018. URL <https://proceedings.mlr.press/v80/haarnoja18b.html>.
- A. H. Hajimiragha and M. R. D. Zadeh. Research and development of a microgrid control and monitoring system for the remote community of bella coola: Challenges, solutions, achievements and lessons learned. In *2013 IEEE International Conference on Smart Energy Grid Engineering (SEGE)*, pp. 1–6, August 2013. DOI: 10.1109/SEGE.2013.6707898.
- Rachid Hassani and Mathieu Lambert. Real-time battery optimization for remote microgrids. Les Cahiers du GERAD G-2024-49, Groupe d’études et de recherche en analyse des décisions, GERAD, Montréal QC H3T 2A7, Canada, August 2024.
- J. Hirwa, O. Ogunmodede, A. Zolan, and A. M. Newman. Optimizing design and dispatch of a renewable energy system with combined heat and power. *Optimization and Engineering*, 23(3): 1–31, September 2022. DOI: 10.1007/s11081-021-09674-4.
- Kai-Chieh Hsu, Haimin Hu, and Jaime Fernández Fisac. The safety filter: A unified view of safety-critical control in autonomous systems. (arXiv:2309.05837), Sep 2023. DOI: 10.48550/arXiv.2309.05837. URL <http://arxiv.org/abs/2309.05837>. arXiv:2309.05837 [cs, eess].
- International Organization for Standardization (ISO). Reciprocating internal combustion engine driven alternating current generating sets — part 1: Application, ratings and performance, 2018. URL <https://www.iso.org/standard/68539.html>. Edition 3.

-
- Nils Jansen, Bettina Könighofer, Sebastian Junges, and Roderick Bloem. Shielded decision-making in mdps. In *arXiv preprint arXiv:1807.06096*, 2018.
- Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.
- Bettina Könighofer, Mohammed Alshiekh, Roderick Bloem, Laura Humphrey, Robert Könighofer, Ufuk Topcu, and Chao Wang. Shield synthesis. *Formal Methods in System Design*, 57:79–115, 2020.
- Hanna Krasowski, Jakob Thumm, Marlon Müller, Lukas Schäfer, Xiao Wang, and Matthias Althoff. Provably safe reinforcement learning: Conceptual analysis, survey, and benchmarking. (arXiv:2205.06750), Nov 2023. DOI: 10.48550/arXiv.2205.06750. URL <http://arxiv.org/abs/2205.06750>. arXiv:2205.06750 [cs].
- Kyung-Bin Kwon and Hao Zhu. Reinforcement learning-based optimal battery control under cycle-based degradation cost. *IEEE Transactions on Smart Grid*, 13(6):4909–4917, Nov 2022. ISSN 1949-3053, 1949-3061. DOI: 10.1109/TSG.2022.3180674.
- M. Lambert and R. Hassani. Diesel genset optimization in remote microgrids. *Applied Energy*, 340: 121036, June 2023. DOI: 10.1016/j.apenergy.2023.121036.
- Yongshuai Liu, Avishai Halev, and Xin Liu. Policy learning with constraints in model-free reinforcement learning: A survey. In Zhi-Hua Zhou (ed.), *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pp. 4508–4515. International Joint Conferences on Artificial Intelligence Organization, 8 2021. DOI: 10.24963/ijcai.2021/614. URL <https://doi.org/10.24963/ijcai.2021/614>. Survey Track.
- Zuxin Liu, Zhepeng Cen, Vladislav Isenbaev, Wei Liu, Zhiwei Steven Wu, Bo Li, and Ding Zhao. Constrained variational policy optimization for safe reinforcement learning, 2022. URL <https://arxiv.org/abs/2201.11927>.
- Kaustubh Mani, Vincent Mai, Charlie Gauthier, Annie S Chen, Samer B. Nashed, and Liam Paull. Risk informed policy learning for safer exploration. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=gJG4IPwg6l>.
- Martin Obermayr, Christian Riess, and Jürgen Wilde. A novel online 4-point rainflow counting algorithm for power electronics. *Microelectronics Reliability*, 120:114112, May 2021. ISSN 00262714. DOI: 10.1016/j.microrel.2021.114112.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL <https://arxiv.org/abs/1707.06347>.
- J. Silvente, G. M. Kopanos, E. N. Pistikopoulos, and A. Espuña. A rolling horizon optimization framework for the simultaneous energy supply and demand planning in microgrids. *Applied Energy*, 155:485–501, October 2015. DOI: 10.1016/j.apenergy.2015.05.090.
- Payam Siyari, Bistra Dilkina, and Constantine Dovrolis. Emergence and evolution of hierarchical structure in complex systems. In *Dynamics On and Of Complex Networks III: Machine Learning and Statistical Physics Approaches 10*, pp. 23–62. Springer, 2019.
- Akifumi Wachi, Xun Shen, and Yanan Sui. A survey of constraint formulations in safe reinforcement learning, 2024. URL <https://arxiv.org/abs/2402.02025>.
- Masaki Waga, Ezequiel Castellano, Sasinee Pruekprasert, Stefan Klikovits, Toru Takisaka, and Ichiro Hasuo. Dynamic shielding for reinforcement learning in black-box environments, 2022. URL <https://arxiv.org/abs/2207.13446>.

-
- Y. Wang, M. Xiao, Y. You, and H. V. Poor. Optimized energy dispatch for microgrids with distributed reinforcement learning. *IEEE Transactions on Smart Grid*, pp. 1–1, 2024. DOI: 10.1109/TSG.2023.3331467.
- Bolun Xu, Alexandre Oudalov, Andreas Ulbig, Goran Andersson, and Daniel S. Kirschen. Modeling of lithium-ion battery degradation for cell life assessment. *IEEE Transactions on Smart Grid*, 9(2):1131–1140, Mar 2018. ISSN 1949-3053, 1949-3061. DOI: 10.1109/TSG.2016.2578950.
- T. Yang, L. Zhao, W. Li, and A. Y. Zomaya. Dynamic energy dispatch strategy for integrated energy system based on improved deep reinforcement learning. *Energy*, 235:121377, November 2021. DOI: 10.1016/j.energy.2021.121377.
- Peipei Yu, Hongcai Zhang, Yonghua Song, Hongxun Hui, and Ge Chen. District cooling system control for providing operating reserve based on safe deep reinforcement learning. *IEEE Transactions on Power Systems*, 39(1):40–52, Jan 2024. ISSN 0885-8950, 1558-0679. DOI: 10.1109/TPWRS.2023.3237888.
- Jin Zhang, Yuxiang Guan, Liang Che, and Mohammad Shahidehpour. Ev charging command fast allocation approach based on deep reinforcement learning with safety modules. *IEEE Transactions on Smart Grid*, 15(1):757–769, Jan 2024. ISSN 1949-3053, 1949-3061. DOI: 10.1109/TSG.2023.3281782.

Supplementary Materials

The following content was not necessarily subject to peer review.

A Agent architecture

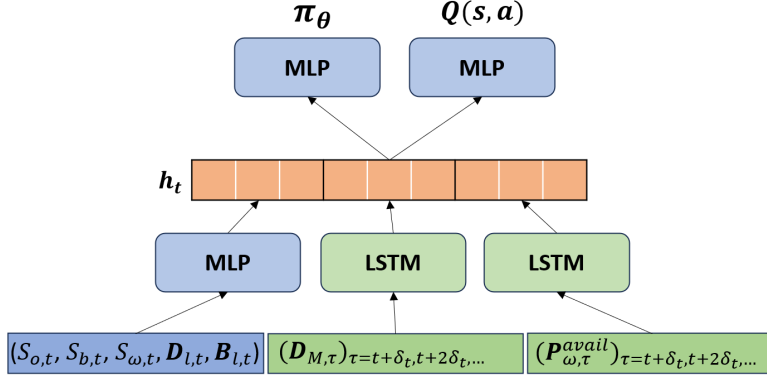


Figure 5: The agent architecture

All RL agents use SAC for decision making. Both the actor and the critic use the same architecture (See Figure 5). The state is formatted as a vector and fed into a multi-layer perceptron (MLP). To take into account the predicted future demand and wind power, we added two LSTM modules which takes the demand and wind power predictions and concatenate their hidden representations with the encoded representation of the state. The final representation is passed through another MLP to compute the action probabilities and state values.

B Learning hyper-parameters

In Table 1, we provide the values for the learning agent and the optimization hyper-parameters.

SAC agent		
Observation encoder	MLP	2 layers (128, 32)
Hidden dimension size	H	3×32
Optimization		
Learning Rate	lr	0.001, 0.0003
Batch size	B	32 , 64

Table 1: Learning Hyper-parameters. The final selected values are in bold.

C Battery Degradation

Different factors can degrade a battery and reduce its storage capacity. Time and temperature affect its capacity, however, these are not under the agent’s control. More relevant here, cycling, i.e. repeated charging and discharging, significantly impacts the battery’s degradation (Cao et al., 2020). Cycle-based battery degradation is often estimated through the rainflow analysis of the battery’s state of charge (SoC) (Xu et al., 2018), which accounts for the number and amplitude of cycles along time. However, this process is usually done offline, given a history of data. In RL however, a degradation cost $d_{b,t}$ at every time step is needed that reflects the impact of the agent’s action while, when summed over a trajectory, being equal to the result of the offline process.

A commonly used approach for online battery degradation estimation is to use a linear approximation, as presented by [Cao et al. \(2020\)](#). The linearized degradation cost is calculated using a simplified linear model, where the cost is directly proportional to the absolute value of the battery SoC change, $b_t = \text{soc}_t - \text{soc}_{t-1}$. The linearized degradation cost is thus computed as $d_{b,t} = \alpha_d \cdot |b_t|$ where α_d is the degradation coefficient. This linear approach provides a straightforward estimation of the degradation cost, making it easier to implement and understand while still capturing the essential impact of battery usage on its lifespan.

In this paper, we aimed at representing the industrial environment as precisely as possible, and as such, to model the battery degradation more accurately. [Kwon & Zhu \(2022\)](#) have proposed an approach to online cycle-based degradation cost estimation for RL. In battery fatigue modeling, the impact of a cycle on degradation is exponential to the cycle amplitude. An interesting insight from [Kwon & Zhu \(2022\)](#) is their decomposition of the cost for each time step as

$$d_{b,t} = \alpha_d \cdot \exp(\beta \cdot |\text{soc}_t + b_t - R[-1]|) - \alpha_d \cdot \exp(\beta \cdot |\text{soc}_t - R[-1]|)$$

where α_d represents the degradation rate, β indicates the sensitivity to SoC changes, and $R[-1]$ is the most recent switching point in the SoC history.

The difficulty of the rainflow algorithm is however to find the right value of $R[-1]$, as cycles can close, and at the next time step, the SoC evolution can pertain to another cycle open further in history. An important problem is that, in theory, the necessary memory to correctly assign each time step to the correct value of $R[-1]$ can be as long as the complete SoC history. [Kwon & Zhu \(2022\)](#) propose an algorithm to keep only three points, however, it does not fully respect rainflow analysis. Instead, we based our approach on [Obermayr et al. \(2021\)](#), where a 4-point online rainflow algorithm allows to identify the last switching point while memory requirement is hard-capped thanks to a discretization process. Our slightly adapted algorithm is summarized in Algorithms 4 and 5. The resulting buffer R is used to compute the reward, and is given as an observation to the agent as part of the battery state to respect the Markovian assumption for the environment.

Note that the discretization can cause artifacts, such as the calculated degradation cost $d_{b,t}$ being negative close to switching points. In this case, we instead approximate it for this time step based on the discretization window w : $d_{b,t} = \alpha_d \cdot (\exp(\beta \cdot |w|) - 1)/w$.

A final question regards the value of α_d and β , so that $d_{b,t}$ represents a realistic capacity loss for the battery. We chose $\beta = 1$ and $\alpha_d = 5$ as in ([Cao et al., 2020](#)). However, these values should be calibrated with a real battery if to be deployed in the real world. This explains our choice to describe the battery degradation units are arbitrary in the paper, and keep the reward function flexible with different values of weight α .

This approach ensures that the degradation cost accurately reflects the impact of battery usage on its lifespan, considering both the magnitude and frequency of usage changes.

Algorithm 4 Rainflow 4-point algorithm

```
// 4-point rainflow condition check
function RAINFLOW4P(R)
    cycleFound  $\leftarrow$  False
    if  $\min(R[-4], R[-1]) \leq \min(R[-3], R[-2])$  and  $\max(R[-3], R[-2]) \leq$ 
     $\max(R[-4], R[-1])$  then
        cycleFound  $\leftarrow$  True
    end if
    return cycleFound
end function

// Main function to update switching point buffer R
function UPDATESWITCHINGPOINTS(x, F, R)
    F[2]  $\leftarrow$  Discretize(x, d_load)
    F, tpFound  $\leftarrow$  HysteresisFilter(F)
    if tpFound then
        Append(R, Discretize(F[0], d_load))
    end if
    Append(R, Discretize(x, d_load))
    while Length(R)  $\geq$  4 and Rainflow4P(R) do
        R[-3]  $\leftarrow$  R[-1]
        Delete(R[-2:])
    end while
    Delete(R[-1])
    return F, R
end function
```

Algorithm 5 Hysteresis Filter algorithm

```
function HYSTERESISFILTER(F)
  // Apply hysteresis filter to the input signal
  // Define helper functions
  function SHIFT(F)
     $F[0] \leftarrow F[1]$ 
     $F[1] \leftarrow F[2]$ 
    return F
  end function
  function SKIP(F)
     $F[1] \leftarrow F[2]$ 
    return F
  end function
   $tpFound \leftarrow False$ 
  if  $F[2] < F[1]$  then
    if  $F[0] \geq F[1]$  then
       $F \leftarrow \text{Skip}(F)$ 
    else
       $tpFound \leftarrow True$ 
       $F \leftarrow \text{Shift}(F)$ 
    end if
  else if  $F[2] > F[1]$  then
    if  $F[0] \leq F[1]$  then
       $F \leftarrow \text{Skip}(F)$ 
    else
       $tpFound \leftarrow True$ 
       $F \leftarrow \text{Shift}(F)$ 
    end if
  else
    if  $F[0] > F[1]$  then
       $F[1] \leftarrow \min(F[1], F[2])$ 
    else
       $F[1] \leftarrow \max(F[1], F[2])$ 
    end if
  end if
  return F,  $tpFound$ 
end function
```
