Emulation of Zonal Controllers for the Power System Transport Problem

Eva Boguslawski, Alessandro Leite, Benjamin Donnot, Matthieu Dussartre, Marc Schoenauer

Emulation of Zonal Controllers for the Power System Transport Problem

Eva Boguslawski[1,2], Alessandro Leite[1], Benjamin Donnot[2], Matthieu Dussartre[2], and Marc Schoenauer[1]

[1] TAU, LISN, INRIA Saclay, Paris-Saclay University
[2] RTE Réseau de Transport d'Electricité

Abstract. Due to the integration of more and more intermittent renewable energy sources, electrical Transmission System Operators (TSOs) embrace a more dynamic grid management, and zonal controllers have emerged as critical components. These controllers, powered by optimization algorithms, oversee specific zones within the power grid, intervening on the power grid to mitigate line overflows. However, their choice of actions and effectiveness depends on predefined target plans provided by human operators, necessitating decision-support tools to streamline this process. Because of reinforcement learning (RL) performance in sequential problems, RL holds promise in devising such plans, yet its reliance on extensive training iterations presents a time-consuming challenge. In this paper, we propose an emulator for zonal controllers, facilitating sufficient training iterations within reasonable computation times. Our methodology combines RL with heuristic techniques tailored to alleviate common challenges encountered in RL applications, such as reducing the action space and incorporating expert knowledge where beneficial. We further introduce a framework for modeling this hybrid approach within a specific Markov Decision Process tailored to the training phase. To systematically tackle the complexities inherent in this problem, we adopt a phased approach, progressively identifying and resolving key challenges. We obtained an emulator that can handle dangerous real-time situations while following the target plan when possible. By integrating RL with domain-specific heuristics and leveraging a structured problem decomposition strategy, our methodology offers a promising avenue for efficiently training decision-support systems for zonal controllers in power grid management.

Keywords: Power grid operation · Reinforcement learning · Target setpoint · Zonal controllers

1 Introduction

TSOs' mission is to ensure safe, reliable energy transmission. Any failure within their grid could impact a large number of customers, leading to economic, personal, and property damages. TSOs handle risks like natural hazards to ensure uninterrupted supply while keeping as low as possible the operational costs.

The massive arrival of renewable energies and the development of energy exchanges with other countries are already changing electricity flows, making them more volatile and less predictable. These new operating conditions require new management strategies for power grid operation. To operate the power grid as close as possible to its limits and thus reinforce its flexibility in real-time, TSOs are developing zonal controllers. A zonal controller monitors a specific zone of the power grid. A zonal controller can handle overflow issues thanks to the predictive control optimization techniques [18]. Likewise, it can limit the production of renewable generators (aka curtailment), control storage units, and (dis)connect power lines. To augment controllers' decision-making, they can be fed targeted plans from human operators, as shown in Figure 1 and Figure 2, addressing potential gaps in their perception of the broader network or problems that might occur in the power grid. Such a plan can delineate desired or undesired grid configurations for the upcoming hours. Consequently, a controller must follow the target plan as closely as possible, but it can deviate from it if a real-time issue forces it to take remedial actions.

Operators have to manage more and more tasks in real-time due to the increase in renewable production. Thus, TSOs need a decision-support assistant to recommend relevant plans and help them manage the power grid. RL [19] is a promising method to elaborate such target plans. However, to achieve good performance, RL requires many training iterations, which is time-consuming. In this context, this paper describes the design of an emulator of a zonal controller fast enough to allow a sufficient number of iterations and with reasonable computation times when training the planner. Indeed, a slightly less realistic but faster emulator in RL can be more efficient than a very accurate but slow one [20].

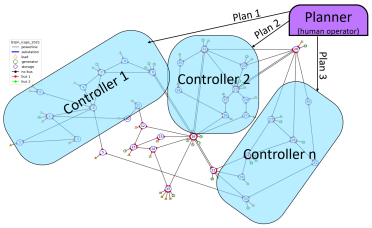

Fig. 1: Each controller (in blue) is responsible for implementing a plan defined by the planner (in purple) in its own zone.

Within this framework, our contributions are twofold: (a) a model to wrap the Markov Decision Process (MDP) to solve and a heuristic into another MDP specific to the training phase. (Section 4.3) (b) an agent managing the power grid while adhering to a curtailment and storage charge plan (Section 5.4).

2 Context and problem definition

2.1 Controllers' mission

Operators have more and more information to process and a growing workload. Some low-level tasks can be automated, while others require human interventions. As explained in Section 1, a controller aims at managing power flow problems automatically in their zone.

However, by construction, a controller alone does not see outside its zone and only projects itself into the next few minutes. In some situations, minimizing the

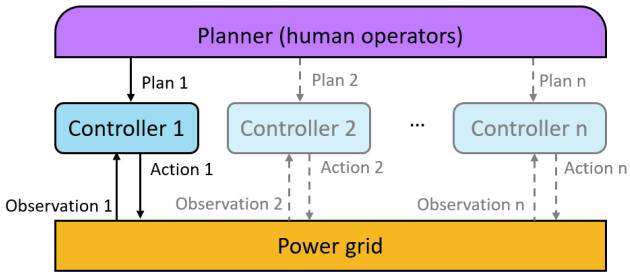

Fig. 2: Illustration of zonal controllers. The planner sends a plan to each controller. Each controller provides an action depending on the received observation and plan. The actions are then applied to the power grid here. Some controllers are grayed because, in our experiments, there is only one controller on the grid.

operational cost requires a more global geographical and temporal view. For this reason, TSOs have introduced the notion of a target plan. For the moment, the plan concerns curtailment and storage units management, but it can be extended to other actions, such as changing the topology of the grid.

2.2 Problem definition

To make operators' work easier, TSO needs a decision-support assistant to recommend relevant target plans. In our paper, this high-level decision-maker is called a planner. RL is a promising method to elaborate such target plans because of its performance on sequential problems. However, RL needs a simulator with which to interact multiple times for training. In the real world, a controller is called every few seconds. A planner has a long-term view (a few hours), so calling a controller every few seconds wouldn't make sense and would be very time-consuming. The aim of this paper is to develop an emulator that reproduces the behavior of a controller on a 5-minute time scale. Accordingly, we want to reproduce the two most important characteristics of a controller: 1) avoiding a blackout at all costs and 2) following the target plan whenever possible. Consequently, we face a multi-objective and high-dimensional optimization problem.

We are especially interested in RL because:

- A deep reinforcement learning (DRL) agent has a fast inference time
- RL performs well with sequential problems since it can take into account uncertainties and future consequences in the long term.
- RL strategies achieved good results during the L2RPN competitions [7]. These competitions aimed to develop an agent operating a simulated power grid in near real-time, a problem similar to the one described in this paper.

We would like to draw attention to the fact that there will be two levels of tasks using RL: (a) the planner level (not discussed here[1]) to coordinate the

[1] The planner is rudimentarily modeled as sending random instructions, see Section 4.1

controllers through target plans and (b) the lower level (subject of this paper) aiming at emulating controllers and facilitating the future training of the planner.

We face different challenges, including

- the curse of our dimensionality [19]: the power grid representation dimension and the number of possible actions are high.
- a multi-objective problem, with one of which takes priority over the other: avoiding blackout at all costs.
- expert knowledge tells us that doing nothing is often the best action when the grid is safe, but it does not solve dangerous situations. This problem is close to the unbalanced data problem in supervised classification.

3 Related Work

Recent years have seen an increasing interest in developing reinforcement learning policies to support the operation of power grids. It is mainly due to the advancement of DRL [5] methods. For example, the winners of the Learning to Run a Power Network (L2RPN) competitions [10] combined deep reinforcement learning approaches and optimization methods to manage a simulated power grid. The RL state-of-the-art algorithm named Proximal Policy Optimization (PPO) [15] especially provided good performance [6]. During the 2022 edition [16,8], continuous actions played an important role for the three better participants (contrary to previous editions). Continuous actions include re-dispatching, curtailment, and actions on storage units. The winners [4,12] successfully manage to use continuous actions, mostly using optimization or sampling to build their actions and do not rely on a training phase.

In [17], the authors proposed an RL-based policy for battery control in a power grid considering grid and financial constraints using an adaptation of the Deterministic Policy Gradient (DDPG) algorithm. Recently, [13] proposed an improved version of the Soft Actor-Critic (SAC) algorithm to optimize battery dispatch over a long horizon.

In this work, we combine reinforcement learning policies with expert rules to learn to operate a power grid while adhering to a plan. We successfully use RL to handle continuous actions such as curtailment and storage power changes. One challenge is to achieve two objectives simultaneously: maximizing survival time and following the plan as closely as possible.

4 Proposed solution: Combining an RL Policy and a heuristic for Zonal Controller Emulation

4.1 Modeling plan following through a Markov Decision Process

A MDP is a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, R, T, \gamma)$ with $\mathcal{S}$ the set of states, $\mathcal{A}$ the set of actions, $\mathcal{T}$ the transition probability function, R the immediate

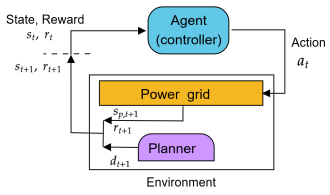

Fig. 3: MDP to solve

reward function, T the length of an episode and γ the discount factor.

Operating a power grid has been modeled as a MDP $\mathcal{M}_p = (\mathcal{S}_p, \mathcal{A}_p, \mathcal{T}_p, R_p, T_p, \gamma_p)$ (with an index p like Power grid) in the grid2op documentation [3]. You can read it for more details, but we will summarize the points that concern us in the next paragraphs. We extend $\mathcal{M}_p$ to $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, R, T_p, 1)$ shown in Figure 3, so that it takes into account a plan.

State space $\mathcal{S} = \mathcal{S}_p \times \mathcal{D}$. The state $s_t = (s_{p,t}, d_t)$ contains:

1. The complete state of the power grid $s_{p,t} \in \mathcal{S}_p$ at time step t. I.e., all information over power nodes (electricity produced and consumed), flows of each power line, storage powers, curtailment limits, date, and more. Like in the real world, any information about the future (such as real consumption or productions a few steps ahead) cannot be observed.
2. The target plan $d_t \in \mathcal{D}$ (sent by the planner) containing:
   - Desirable storage charges at $t+1$. Remember that the agent controls the storage powers but not directly their charges.
   - Desirable *curtailment* limits (maximum limits applied to each renewable production) at $t+1$. In our case, the desirable limits are always 100% of the maximum possible productions i.e., we curtail as little as possible.

   As a result, $\mathcal{D} = (\times_{i=1,\dots,N_{storage}}[0, E_{max,i}]) \times \{100\}^{N_G}$ where $N_{storage}$ is the number of storage units, $E_{max,i}$ is the maximum charge of the storage unit i and N_G is the number of renewable generators.

Action space $\mathcal{A} \subset \mathcal{A}_p$. The action space contains both discrete actions and continuous actions: 1. Line connection or disconnection. 2. Curtailment on renewable generators. 3. Storage powers setpoint change.

Transition probability function . The transition function $\mathcal{T}_p$ relies on the electrical grid equations. We need to complete $\mathcal{T}_p$ to include the target plan behavior in the model. Since the planner design is not covered in this paper, we suppose the target plan always follows a constant and independent probability law $\mathcal{L}_{planner} \in \Delta(\mathcal{D})$. Then, we define $\mathcal{T}$ such that:

$$\mathcal{T}(s_t, a_t, s_{t+1}) = \mathcal{T}_p(s_{p,t}, a_t, s_{p,t+1}) \times p_{\mathcal{L}_{planner}}(d_t, d_{t+1})$$

We also want to mention that, in $\mathcal{M}_p$, the blackout state s_p^{done} is an *absorbing state*. As explained in [19], an absorbing state only transitions to itself and generates only rewards of zero. You can see it like a game over state. A blackout is triggered if total demand is not met anymore, most of the time because of too many overflows on lines. Once you have reached a blackout in an episode, you cannot get out of it.

Reward. We design rewards used for evaluation. We have two different goals.

1. Avoiding blackout as long as possible. It leads to the following reward:

$$R_{Survival}(s_t, a_t, s_{t+1}) = \begin{cases} 1 \text{ if } s_{p,t+1} \neq s_p^{done} \\ 0 \text{ otherwise} \end{cases}$$

2. Following the plan along the episode. We design two rewards $R_{Curtailment}$ and $R_{StorageCharge}$ that penalize the distance to the plan:
   - Let be $l_{g,t}$ the observed curtailment limit (part of $s_{p,t}$) of the renewable generator g at time step t normalized by the maximum possible production of the generator g. Let be N_G the number of renewable generators. As we explained, the curtailment plan is always 100%, so we define:

$$R_{Curtailment}(s_t, a_t, s_{t+1}) = \begin{cases} \frac{1}{N_G} \sum_{g=1}^{N_G} l_{g,t+1} \text{ if } s_{p,t+1} \neq s_p^{done} \\ 0 \qquad\qquad\qquad\quad \text{otherwise} \end{cases}$$

   - We want a storage charge reward that penalizes the absolute differences between the observed storage charges and the target storage charges normalized by the maximum storage charges. Let $c_{obs,i,t}$ be the observed charge (part of $s_{p,t+1}$) and $c_{target,i,t}$ the target charge (part of d_t) of the storage unit i at time step t (MWh).

$$d_{StorageCharge}(s_t, a_t, s_{t+1}) = \frac{1}{N_{storage}} \sum_{i=1}^{N_{storage}} \frac{|c_{obs,i,t+1} - c_{target,i,t}|}{E_{max,i}}$$

Now we can define:

$$R_{StorageCharge}(s_t, a_t, s_{t+1}) = \begin{cases} 1 - d_{StorageCharge}(s_t, a_t, s_{t+1}) \text{ if } s_{p,t+1} \neq s_p^{done} \\ 0 \qquad\qquad\qquad\qquad\qquad\qquad\quad \text{otherwise} \end{cases}$$

By combining the two goals' rewards, we end up with:

$$\begin{aligned} R(s_t, a_t, s_{t+1}) = \alpha_1 R_{Survival}(s_t, a_t, s_{t+1}) + \alpha_2 R_{Curtailment}(s_t, a_t, s_{t+1}) \\ + \alpha_3 R_{StorageCharge}(s_t, a_t, s_{t+1}) \end{aligned}$$

4.2 Simplification of the problem

The MDP $\mathcal{M}$ presented in Section 4.1 is experimentally hard to solve. Finding effective rewards and hyper-parameters for the training part can be very challenging in RL. Indeed, using a different reward R' during the training can sometimes lead to better performance. For this reason, we decided to first work on two simplified versions of $\mathcal{M}$:

- $\mathcal{M}_1$ focusing on operating the power grid described in Section 5.2
- $\mathcal{M}_2$ focusing on following a storage charge plan described in Section 5.3

These works have enabled us to design suitable rewards for the training parts and to find appropriate hyper-parameters (see Section 5.1). We then tackle $\mathcal{M}$ in Section 5.4 by combining the designed training reward and using the previously identified hyper-parameters.

4.3 Wrapping a heuristic into a Markov decision process

Previous works showed that RL agents can reach better performances when associated with heuristics [21],[1],[14]. Indeed, simple expert rules maintain the power grid well in most situations. For example, to operate a power grid, doing nothing is a very wise action when the grid is safe (most time steps). It can also reduce the action space. We call heuristic hard-coded expert rules that provide an action that is supposed to be used instead of the PPO policy in specific situations.

We model a heuristic with the tuple $h = (\alpha, h_{expert}, h_{check})$ with $\alpha : \mathcal{S} \to \{0, 1\}$, $h_{expert} : \mathcal{S} \to \mathcal{A}$ and $h_{check} : \mathcal{S} \times \mathcal{A} \to \mathcal{A}$. The function α returns 0 when the PPO policy π_{PPO} needs to be called and 1 when we stick to the chosen expert rules. The function h_{expert} returns the relevant action to take according to the expert rules. The function h_{check} aims at adding minor changes to the action proposed by π_{PPO} if needed. Now, we can design agents with this formalization:

$$a(s) = \begin{cases} h_{check}(s, \pi_{PPO}(s)) & \text{if } \alpha(s) = 0 \\ h_{expert}(s) & \text{otherwise} \end{cases}$$

Note that, by adding a heuristic, the MDP to solve is different from $\mathcal{M}$ when training π_{PPO}. Indeed, when the heuristic is applied to the environment, several intermediate transitions (performed by h_{expert}) can be "hidden" in one unique transition for π_{PPO} during training. The rewards obtained through these transitions are summed and sent back by the environment. The reward is now stochastic because it depends on these intermediate transitions. This new MDP is $\mathcal{M}^h = (\mathcal{S}, \mathcal{A}, p^h, T_p, 1)$ where p^h is the law of the function F^h defined in 1. F^h samples s^h_{t+1}, r^h_{t+1} given s^h_t, a^h_t. Also, to be able to write F^h with a terminal condition consistent with $\mathcal{M}$, we need to make the number of transitions performed by the agent (including those with h_{expert}) available in s (called $t_{\mathcal{M}}$). Otherwise, we can't know when the episode has to end.

4.4 Evaluation protocol

Firstly, we want to insist that the criteria studied during evaluation don't exactly match what is optimized during the training phase. The goal is to solve the MDP $\mathcal{M}$ and not $\mathcal{M}^h$. Also, if it facilitates training, the reward used during training may differ from the one defined in $\mathcal{M}$.

The stochasticity of the transition function $\mathcal{T}$ mainly stems from the production and consumption. To fairly evaluate our agents, we evaluate them on a test set of N_S scenarios where a scenario means realistic simulated productions and consumption time series that respect a temporal consistency. For a scenario e and an agent a we define:

Algorithm 1: A stochastic function F^h that samples s^h_{t+1}, r^h_{t+1} given a state s^h_t, an action a^h_t and the heuristic h

Input: State s, Action a, Heuristic $h = (\alpha, h_{check}, h_{expert})$
Output: State s', Reward r
Function `Apply(`s, a, h`):`
  if $(t_{\mathcal{M}}$ *in* $s) = T$ **then**
    `/* Case of an absorbing state */`
    $s', r, \alpha, a' \leftarrow s, 0, 0, \varnothing$
  else
    Sample $s' \sim \mathcal{T}(s, a, .)$
    $r \leftarrow R(s, a, s')$
    $\alpha, a' \leftarrow \alpha(s'), h_{expert}(s')$
  return s', r, α, a'
Function F^h`(`s, a`):`
  $a \leftarrow h_{check}(s, a)$
  $s, r_{sum}, \alpha, a' \leftarrow$ Apply(s, a, h) ; `// Action a is applied to s`
  while $\alpha = 1$ **do**
    $s, r, \alpha, a' \leftarrow$ Apply(s, a', h) ; `// Heuristic h can take over`
    $r_{sum} \leftarrow r_{sum} + r$
  return s', r_{sum}

$$T_{end}(e, a) = \sum_{t=0,\dots,T} R_{Survival}(s_t, a(s_t), s_{t+1})$$

$$D_{StorageSetpoint}(e, a) = \frac{1}{T_{end}(e, a)} \sum_{t=0,\dots,T_{end}(e,a)-1} d_{StorageCharge}(s_t, a(s_t), s_{t+1})$$

$$CL(e, a) = \frac{1}{T_{end}(e, a)} \sum_{t=0,\dots,T_{end}(e,a)-1} R_{Curtailment}(s_t, a(s_t), s_{t+1})$$

For an agent a, we note $\overline{T}_{end}(a), \overline{D}_{StorageCharge}(a), \overline{CL}(a)$ the average of these indicators over the N_S test scenarios.

The priority for the agent is to avoid blackouts as long as possible. Therefore, the main criterion to evaluate an agent a on our test scenarios is the average survival time $\overline{T}_{end}(a) \in [\![1, \dots, T]\!]$. We want it to be as long as possible. In the second stage, we consider two criteria to evaluate the proximity to the plan:

- the average distance to the storage charge setpoint given by $\overline{D}_{StorageCharge}(a)$. We want it to be low (close to 0).
- the average curtailment limit given by $\overline{CL}(a)$. We want it to be high (close to 100%).

Moreover, the criterion $\overline{T}_{end}$ has priority over others because if the plan endangers the grid, the agent must be able to deviate from it. For this reason, we cannot merge our indicators into a single reward during the evaluation phase.

5 Experimental results

5.1 Common experimental setup

We use the PPO algorithm[2] because of its good results on a similar planning problem [6]. Moreover, PPO is a popular algorithm in RL because of its good sample efficiency, its ease of implementation, and its ease of tuning. Through our experiments on the MDP $\mathcal{M}_1$, we calibrated, among others, the policy's neural network architecture and the learning rate. As for the MDP $\mathcal{M}_2$, it has proved to be sensitive to the rollout buffer size. The hyper-parameters used to train the PPO policy are available in our code[3]. This set of hyper-parameters is interesting because, in our experiments, it has been robust to neural networks' initialization changes and even to reward changes.

The environment is a simulated but realistic power grid with 118 substations based on the IEEE-118 grid[4]. A time step is equivalent to 5 minutes. The agent can control 186 lines, 42 renewable generators (N_G), and 7 storage units ($N_{Storage}$). i.e., 2^{186} topological actions are available, representing continuous actions in a 49-tuple.

Each scenario lasts a week ($T = 2016$). The scenarios[5] are split into a training set (64 scenarios) and a test set (34 scenarios). The results presented in this paper are all computed on the test set.

5.2 Operating the power grid without a plan

As explained in Section 4.2, in this section, we focus on training an agent that can operate a power grid without having a plan to follow.

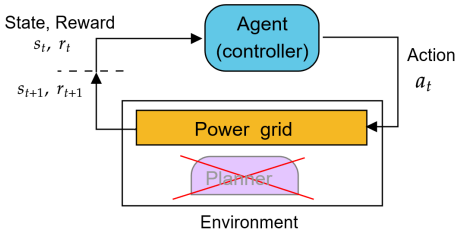

Fig. 4: Same diagram than in Figure 3 except that we remove the planner.

Table 1: Survival time $\overline{T}_{end}$ (average and median over the 6 trained agents)

Agent	Survival time $\overline{T}_{end}$ (avg ± std)
DoNothing	549 / 2017
RecoPowerline	471 / 2017
NoPlan agent	**1404 ± 66** / 2017

Experimental settings We end up with a simplified version of the initial MDP $\mathcal{M}_1 = (\mathcal{S}_1, \mathcal{A}_1, \mathcal{T}_1, R_1, T, 1)$ with: $\mathcal{S}_1 = \mathcal{S}_p$, $\mathcal{A}_1 = \mathcal{A}$, $\mathcal{T}_1 = \mathcal{T}_p$, and $R_1 = R_{Survival}$.

We trained 6 agents over 10 million iterations[1] with a learning rate of $3e^{-5}$. A training lasts about a day.

[2] We used the `stable-baselines3` [11] and `grid2op` [2] Python libraries.
[3] The code is available at `tinyurl.com/4wr5bmaz`
[4] The IEEE 118-bus grid is a portion of the American Electric Power system in 1962, available at `tinyurl.com/yabtkbru`
[5] We generate scenarios with the `chronix2grid` [9] Python library

Training reward. The reward used to train the PPO policy is linear with line capacity usages. It returns 1 if no current is flowing in the lines. It returns 0 if all lines are used at their maximum capacities. This quantity can be seen as an indicator of the grid safety. Let $\rho_l(s)$ be the capacity of the connected power line l of state s where the capacity is the observed current flow on l divided by the thermal limit of the line (if $\rho(s)_l \geq 1$ the line l is in overflow). Let $N_L(s)$ be the number of connected lines at state s). Then $R_1^{Training}(s_t, a_t, s_{t+1}) = R_{LinesCapacityReward}(s_t, a_t, s_{t+1})$ with:

$$R_{LinesCapacityReward}(s_t, a_t, s_{t+1}) = 1 - \frac{1}{N_L(s)} \sum_{l=1}^{N_L(s)} \rho_l(s_{t+1})$$

Heuristic. This heuristic handles the discrete actions (line (dis)connection) to reduce the action space of the PPO policy. It also incorporates expert knowledge to solve safe situations to allow the PPO policy to focus on risky situations like in [21].

We define $L(s) = \{l \in [\![1, N_{Lines}]\!], \text{ s.t. line } l \text{ can be reconnected}\}$ the set of reconnectable lines at state s, $l(s) = min\ L(s)$ (an arbitrary way of chosing a line in $L(s)$). and $\rho_{max}(s) = \max_l \rho_l(s)$. Our heuristic $h = (\alpha, h_{expert}, h_{check})$ (see Section 4.3) gathers several objectives:

- **Reconnecting all possible power lines.** Some lines may have been disconnected because of an overflow for example. It stems from a real operational criterion whose aim is to avoid dangerous situations where several lines are disconnected.
- **Doing nothing, if possible.** If $\rho_{max}(s_t)$ is low ($\rho_{max}(s_t) \leq 90\%$), lines are probably safe, so we choose the do-nothing action a_{DN} (safe situation), else we need to call the PPO policy (unsafe situation). Moreover, doing nothing when the grid is safe is generally consistent with what an operator would do in real life. The 90% threshold is a hyper-parameter that can be adjusted. We define:

$$h_{RP}(s) = \begin{cases} a_{ReconnectLine\ l(s)} & \text{if card } L(s) > 0 \\ a_{DN} & \text{otherwise and if } \rho_{max}(s) \leq 90\% \end{cases}$$

- **Limiting action impact.** It limits the curtailment/storage unit power to ensure that the amount of power curtailed/taken to-from the storage units can be compensated by controllable generators.

It gives us $h_{RecoPowerline} = (\alpha_{RP}, h_{RP}, h_{check})$ with:

$$\begin{aligned} \alpha_{RP}(s) &= \begin{cases} 1 \text{ if card } L(s) > 0 \text{ or } \rho_{max}(s) \leq 90\% \\ 0 \text{ otherwise} \end{cases} \\ h_{check}(s) &= LimitActionImpact(s) \end{aligned}$$

Results We compare our trained agent with two expert baselines Table 1:

- **"DoNothing" agent**: $\forall s,\ a_{DoNothing}(s) = a_{DN}$
- **"RecoPowerline" agent**: $\forall s,\ a_{RecoPowerline}(s) = h_{RP}(s)$

Our agent survives on average more than twice as long as the "DoNothing" agent and three times longer than the "RecoPowerline" agent.

5.3 Following the plan without operating the power grid

We now focus on the second difficulty, which is following a storage charge setpoint (sent by the planner) by controlling the storage powers as illustrated Figure 5.

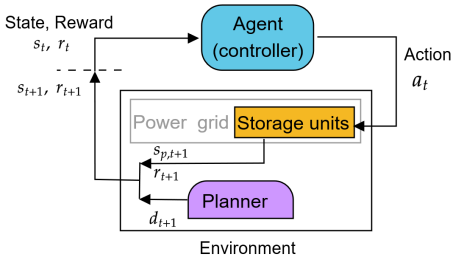

Fig. 5: Same diagram than in Figure 3 except that we ignore the grid safety aspect.

Table 2: Average distance to storage setpoint $\overline{D}_{StorageCharge}$ (without retraining)

Target setpoint shape	Trained agent	Expert agent
Smooth	1.2e-2	3.0e-3
Noisy	1.7e-2	1.1e-2
Step function	2.9e-2	1.4e-2

Experimental setting As explained in Section 4.2, in this part, we ignore the safety of the power grid aspect. Thus, we study a simplified version of the initial MDP $\mathcal{M}_2 = (\mathcal{S}_2, \mathcal{A}_2, \mathcal{T}_2, T_2, 1)$ shown in 5. We have $R_2 = R_{StorageCharge}$ and :

- $\mathcal{S}_2$ a state contains only the current storage charges $(c_{obs,i})_{i=1,\dots,N_{storage}}$ and the desirable storage charges for the next time step $(c_{target,i})_{i=1,\dots,N_{storage}}$.
- $\mathcal{A}_2$ is limited to storage powers changes.
- $\mathcal{T}_2$ is a simplification of $\mathcal{T}$. It is such that with $a_t = (p_{i,t})_i$ where $p_{i,t}$ is the chosen power of the storage unit i at step t, we have

$$c_{obs,i,t+1} = \text{clip}(c_{obs,i,t} + \frac{5min}{60min} p_{i,t}, 0, E_{max,i}),\ c_{target,t+1} \sim \mathcal{L}_{plan}(c_{target,t})$$

Training reward. The reward is scaled so that an agent acting randomly gets a reward of 0 and an agent following perfectly the target a reward of 1:

$$R_2^{Training}(s_t, a_t, s_{t+1}) = \begin{cases} 1 - 2\, d_{StorageCharge}(s_t, a_t, s_{t+1}) & \text{if } s_{p,t+1} \neq s_p^{done} \\ 0 & \text{otherwise} \end{cases}$$

During our experiments, we observed that a difference in absolute value gives better results than a squared difference. The scaling operation (in which the penalization is increased) also considerably improves the results.

Heuristic. We don't use any heuristic in this experiment.

Hyper-parameters. We trained 1 agent over 20 million iterations. We use a learning rate of $3e^{-6}$. We need to reduce the rollout buffer size to 32 to be able to learn. This RL problem seems to be particularly sensitive to these two latter hyper-parameters and to the reward function design.

Baseline We compare our trained agent to an **expert agent**. The expert agent computes a storage power (MW) to get as close as possible to the target charge (MWh). The computed power is then clipped with the minimum and maximum power allowed $p_{min,i}, p_{max,i}$. Let $p_{expert,i}(s)$ be the power storage chosen by the expert agent for the storage unit i for the state s: $\forall i = 1, ..., N_{storage}$ $p_{expert,i}(s) = \text{clip}\left((c_{target,i}(s) - c_{obs,i}(s)) \times \frac{60min}{5min},\ p_{min,i},\ p_{max,i}\right)$

Then our expert agent $a_{StoragePlan}(s) = a_{\text{ask } p_{expert}(s) \text{ to storage units}}$

Results Robustness with regards to the shape of the setpoint

To test our agent's ability to generalize, we choose to evaluate it with three kinds of setpoint trajectory shapes:

- **Random smooth shape**: $\mathcal{L}_{planner}^{(smooth)}$ is supposed to be easy to follow. We train the agent with this kind of shape.
- **Random noisy shape**: $\mathcal{L}_{planner}^{(noisy)}$ is supposed to be harder to follow.
- **Random step function shape**: $\mathcal{L}_{planner}^{(stepfunction)}$ is impossible to follow because the heights of the steps are greater than the maximum power the storage unit can produce in a single time step. A solution is to activate maximum (or minimum) power until linearly reaching the target charge of the corresponding step.

To assess the performance of our trained agent, we evaluated it on two different conditions:

1. the target setpoint trajectory is drawn from the same distribution as the one used during the training: $\mathcal{L}_{planner}^{(evaluation)} = \mathcal{L}_{planner}^{(training)} = \mathcal{L}_{planner}^{(smooth)}$
2. we use (without retraining) completely different distributions to those used to train. No retraining of the agent has been done for this evaluation. $\mathcal{L}_{planner}^{(evaluation)} = \mathcal{L}_{planner}^{(noisy)}$ or $\mathcal{L}_{planner}^{(stepfunction)}$

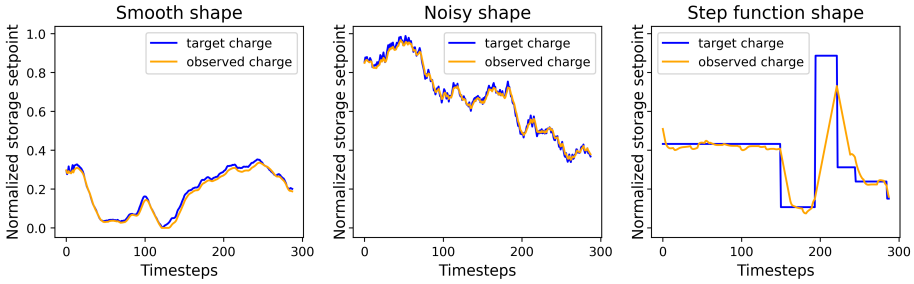

Fig. 6: Examples of three kinds of setpoint trajectories (blue curves) for storage units. The associated observed charge trajectories (orange curves) follow them very closely. It shows that our agent can easily follow a set of various setpoint trajectories.

Examples of setpoint trajectories are shown in Figure 6 with the associated observed charge trajectories. Results are shown in Table 2, we compute

$\overline{D}_{StorageCharge}$. We can visually and quantitatively observe that our trained agent manages to stay very close to the setpoint trajectories and is almost as good as the expert agent and that these results are robust to the shape of the setpoint trajectory. In the case of a step function shape, we observe linear portions during which the agent activates maximum/minimum power. This test is important because, in reality, the setpoint shape can completely change. Being robust to the setpoint shape is important to avoid the need for constant retraining.

Robustness with regards to the storage units characteristics

Changing storage units characteristics is an interesting generalization property, especially if a different storage unit is added to the grid. We can see in Figure 7 performances when increasing or decreasing the storage maximum powers without retraining. We observe that our agent can manage considerable characteristic changes: the maximum powers can be up to 10 times lower/higher and achieve good performance.

- If we decrease the storage maximum powers: as the agent learned to activate maximum powers when needed, it performs almost as well as the expert.
- If we increase the storage maximum powers: up to a certain increase, the agent can compensate by doing little undulations.

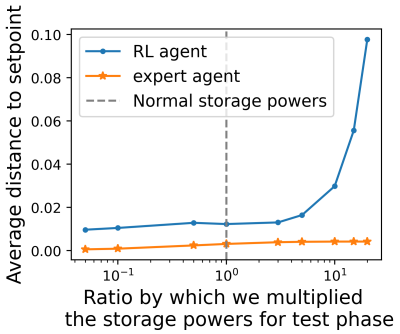

Fig. 7: We multiply the storage maximum powers by a coefficient shown on the x-axis (log scale) and evaluate our agent on the test 34 setpoint trajectories.

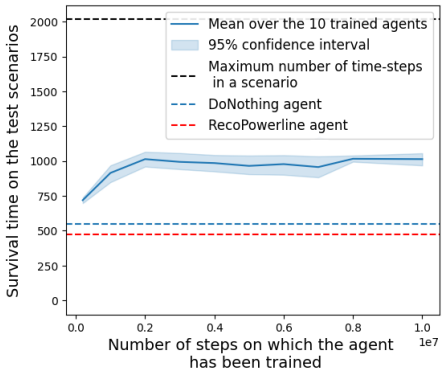

Fig. 8: Learning curve on the test set with 6 instances of the WithPlan agent

5.4 Emulation of zonal controller

We combine the experiments described in Section 5.2 and Section 5.3 to solve the MDP $\mathcal{M}$ described in Section 4.1.

Experimental setting We trained 6 agents over 10 million iterations[1]. We use a learning rate of $3e^{-5}$ like in Section 5.2. A training lasts about a day.

Table 3: Comparing the WithPlan strategy with the baselines

Agent	Reward Weights	Training Heuristic	Evaluation Heuristic	$\overline{T}_{end}$	$\overline{CL}^{\star}$	$\overline{D}_{Storage\ Setpoint}$
DoNothing	–	–	–	549	100%	–
RecoPowerline	–	–	–	471	100%	–
NoPlan agent (Section 5.2)	$\lambda_1 = 1$, $\lambda_2 = 0$, $\lambda_3 = 0$	$h_{Reco\ Powerline}$	$h_{Reco\ Powerline}$	**1404 ± 66**	32% ± 3%	30% ± 2%
NoPlan agent (Section 5.2)	$\lambda_1 = 1$, $\lambda_2 = 0$, $\lambda_3 = 0$	$h_{Reco\ Powerline}$	$h_{WithPlan}$	828 ± 87	68% ± 4%	5% ± 0.3%
Light-Curtailment agent	$\lambda_1 = 0.5$, $\lambda_2 = 0.5$, $\lambda_3 = 0$	$h_{Reco\ Powerline}$	$h_{Reco\ Powerline}$	1281 ± 177	50% ± 2%	30% ± 1%
WithPlan agent	$\lambda_1 = 0.5$, $\lambda_2 = 0.5$, $\lambda_3 = 0.25$	$h_{WithPlan}$	$h_{WithPlan}$	1013 ± 61	**71% ± 2%**	**2% ± 0.1%**

*Curtailment limits

Training reward. We combine the previous rewards $R_1^{Training}$ and $R_3^{Training}$ and add a term to penalize the distance to the target plan for curtailment:

$$R_3^{Training} = \lambda_1 R_{LinesCapacityReward} + \lambda_2 R_{Curtailment} - \lambda_3 d_{StorageCharge}$$

Heuristic. $h_{RecoPowerline}$ (Section 5.2) allows the agent to focus its learning on risky situations. To use the same idea while taking into account the plan, we added new expert rules to the initial heuristic $h_{RecoPowerline}$ to best follow the storage charges / curtailment plan during safe situations.

Expert rule to get closer to the storage charges plan. We follow perfectly the storage charges plan using the baseline $a_{StoragePlan}$ detailed in Section 5.3.

Expert rule to get closer to the curtailment plan. As a reminder, the curtailment plan asks to curtail as little as possible. Yet, the NoPlan agent designed in Section 5.2 highly relies on curtailment actions. So, increasing curtailment limits is much trickier and needs to be done gradually. For this reason, we introduce hyper-parameters (in bold) in our expert rule to modulate the curtailment reduction. We tuned them (grid-search) by combining this expert rule with the agent designed in Section 5.3. We proceed in the following way:

1. **Cancel the useless curtailment limits:** If the production of a generator (normalized by its maximum) is beyond its curtailment limit, we reset its limit to 100% through the action $a_{ResetUselessCurtailment}(s)$.
2. **Increase the useful curtailment limits if possible:** If $\rho_{max}(s_t) \leq$ **85%**, we increase the curtailment limits of the **1** generator(s) with the **higher** curtailed powers by **5%** of their maximum productions. We call this action $a_{SoftenLimits}(s)$.

We obtain the WithPlan heuristic $h_{WithPlan} = (\alpha_{WP}, h_{WP}, h_{check})$ with:

$$\alpha_{WP}(s) = \begin{cases} 1 \text{ if card } L(s) > 0 \text{ or } \rho_{max}(s) \leq 90\% \\ 0 \text{ otherwise} \end{cases}$$

$$h_{WP}(s) = \begin{cases} a_{ReconnectLine\ l(s)} \text{ if card } L(s) > 0 \\ a_{StoragePlan}(s) \quad \& \ a_{ResetUselessCurtailment}(s) \\ \qquad\qquad \& \ a_{SoftenLimts}(s) \\ \qquad\qquad \text{otherwise and if } \rho_{max}(s) \leq 85\% \\ a_{StoragePlan}(s) \quad \& \ a_{ResetUselessCurtailment}(s) \\ \qquad\qquad \text{otherwise and if } \rho_{max}(s) \leq 90\% \end{cases}$$

Results. A great survival time with moderate curtailments

The WithPlan heuristic is efficient. We obtain a stable learning curve shown in Figure 8. By changing the heuristic to "WithPlan" during the evaluation of the NoPlan agent, one can observe in Table 3 that the WithPlan heuristic considerably increases the average curtailment limits and gives an almost zero average distance to the storage setpoint when associated with the NoPlan agent. The drawback is a drop in the average survival time. This can be improved by including the WithPlan heuristics during training. Indeed, Table 3 shows that our WithPlan agent manages to survive for a long time despite using the heuristic.

Choice of $\lambda_1, \lambda_2, \lambda_3$ and final performances. We calibrate the reward's weights by hand to obtain a reward between 0 and 1. We also need the reward to remain positive; otherwise, the agent learns to cause a blackout to end the episode and avoid negative returns. $\lambda_1 = 0.5$ and $\lambda_2 = 0.5$ allow to obtain a high average survival time and a consequent reduction of curtailment limits. We can observe this by comparing the NoPlan agent and the LightCurtailment agent (both with $h_{RecoPowerline}$) in the Table 3. On the contrary, λ_3 is not easy to tune. Large values degrade the survival time $\overline{T}_{end}$, which is our most important criterion. For this reason, we use $\lambda_3 = 0.25$, a pretty small value.

We finally obtain an agent with an average survival time much higher than our initial baselines' times (the "DoNohting" and "RecoPowerline" agents) and quite close to the time of the agent designed in Section 5.2. The design of our reward and our heuristic allows much higher curtailment limits and a close following of the storage charges plan.

The WithPlan heuristic is too efficient: Our agent's PPO policy learns to maintain the storage charges regardless of the storage setpoint. Following the storage charges plan is entirely handled by the WithPlan heuristic. Since the PPO policy is called only if $\rho_{max}(s_t) > 90\%$, the PPO policy is called only for brief moments compared with the whole duration of an episode. The penalties received through the PenalizationStorage during these brief moments are probably too small. In addition, the storage charges setpoint remains relatively stable during these brief moments, so maintaining the storage charges constant is a lazy but efficient way to limit the PernalizationStorage term.

Generalization Our agent cannot be used on a different power grid. Our MDP $\mathcal{M}$ depends entirely on the architecture and characteristics of the power grid

(e.g. its thermal limits). However, the approach presented in this paper can be applied to any power grid.

6 Conclusion

In this paper, we introduce a reinforcement learning (RL) framework aimed at emulating a zonal controller within a power grid environment. Our approach centers on developing an RL agent that can balance the dual objectives of longevity—specifically, avoiding blackout conditions—and adherence to a predefined operational plan. A notable feature of our strategy is the integration of a heuristic to allow our agent, which is based on the PPO algorithm, to focus its learning on unsafe situations. We propose a formalization to combine the MDP to solve and a heuristic into another MDP relevant to the training phase.

Experimental results show that our policy equips the agent with the ability to ensure operational longevity while successfully following a curtailment and storage charging plan across a variety of power grid scenarios. Nonetheless, our findings also reveal a potential limitation: our heuristic may remove curtailment limits too quickly during safe situations, reducing longevity.

To address this limitation and enhance robustness, we propose to explore a multi-agent setup featuring safe/unsafe-situation specialized agents, each with distinct roles. Moreover, we assumed that a unique zonal controller monitors the grid, but in the longer term, we intend to work with several zonal controllers on the grid (like in Figures 1 and 2). The development of an intelligent target plan recommender also needs to be addressed to assist operators.

References

1. Chen, S., Duan, J., Bai, Y., Zhang, J., Shi, D., Wang, Z., Dong, X., Sun, Y.: Active power correction strategies based on deep reinforcement learning—part ii: A distributed solution for adaptability. CSEE Journal of Power and Energy Systems **8**(4), 1134–1144 (2022)
2. Donnot, B.: Grid2Op. github.com/rte-france/grid2op (2020)
3. Donnot, B.: Dive into grid2op sequential decision process. tinyurl.com/38tjysnx (2024)
4. Dorfer, M., Fuxjäger, A.R., Kozak, K., Blies, P.M., Wasserer, M.: Power grid congestion management via topology optimization with alphazero (2022)
5. François-Lavet, V., Henderson, P., Islam, R., Bellemare, M.G., Pineau, J.: An introduction to deep reinforcement learning. Foundations and Trends in Machine Learning **11**(3-4), 219–354 (2018)
6. Hu, Y., Chen, B., Tang, K.: Learning to Run a Power Network (L2RPN) - Robustness Track (2021), bit.ly/46cXhcz
7. L2RPN: Learning to run a power network competition series. l2rpn.chalearn.org (2019)
8. L2RPN: L2RPN 2022 energies of the future and carbon neutrality. codalab.lisn.upsaclay.fr/competitions/5410 (2022)
9. Marot, A., Megel, N., Renault, V., Jothy, M.: ChroniX2Grid - The Extensive PowerGrid Time-serie Generator. https://github.com/BDonnot/ChroniX2Grid (2020)

10. Marot, A., Donnot, B., Romero, C., Donon, B., Lerousseau, M., Veyrin-Forrer, L., Guyon, I.: Learning to run a power network challenge for training topology controllers. Electric Power Systems Research **189**, 106635 (2020)
11. Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., Dormann, N.: Stable-baselines3: Reliable reinforcement learning implementations. Journal of Machine Learning Research **22**(268), 1–8 (2021)
12. Research, A.: L2RPN 2022 solution (2022), tinyurl.com/49y3ars7
13. Sage, M., Staniszewski, M., Zhao, Y.F.: Economic battery storage dispatch with deep reinforcement learning from rule-based demonstrations. In: International Conference on Control, Automation and Diagnosis. pp. 1–6 (2023)
14. van der Sar, E., Zocca, A., Bhulai, S.: Multi-agent reinforcement learning for power grid topology optimization. arxiv:2310.02605 (2023)
15. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv:1707.06347 (2017)
16. Serré, G., Boguslawski, E., Donnot, B., Pavão, A., Guyon, I., Marot, A.: Reinforcement learning for energies of the future and carbon neutrality: a challenge design. arXiv:2207.10330 (2022)
17. da Silva André, J., Stai, E., Stanojev, O., Hug, G.: Battery control with lookahead constraints in distribution grids using reinforcement learning. Electric Power Systems Research **211**, 108551 (2022)
18. Straub, C., Olaru, S., Maeght, J., Panciatici, P.: Zonal congestion management mixing large battery storage systems and generation curtailment. In: IEEE Conference on Control Technology and Applications. pp. 988–995 (2018)
19. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, 2 edn. (2018)
20. Truong, J., Rudolph, M., Yokoyama, N., Chernova, S., Batra, D., Rai, A.: Rethinking sim2real: Lower fidelity simulation leads to higher sim2real transfer in navigation (2022)
21. Yoon, D., Hong, S., Lee, B.J., Kim, K.E.: Winning the l2rpn challenge: Power grid management via semi-markov afterstate actor-critic. In: International Conference on Learning Representations (2021)