POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Distributed Reinforcement Learning for Power Grid Operations

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

Davide Beretta, 10656835

Advisor:
Prof. Marcello Restelli

Co-advisors:
Gianvito Losapio
Marco Mussi
Prof. Alberto Maria Metelli

Academic year:
2023-2024

Abstract: The increasing share of renewables in the energy mix and the globalization of the energy market introduce new challenges in the problem of controlling power grids. The recent series of competitions Learning To Run a Power Network have promoted the use of Reinforcement Learning to aid human dispatchers in managing power grids. So far the proposed solutions significantly limit the action space and rely on either a single agent overseeing the entire grid or multiple independent agents operating at the substation level. In this work, we propose a different approach in which we first decompose the problem by means of a domain-agnostic algorithm that estimates correlations between state and action components entirely based on data. Highly correlated state-action pairs are grouped together to create simpler, possibly independent sub-problems. On this decomposition we can run a distributed learning processes in which each agent interacts with its own sub-problem, reaching a partial solution with reduced computational and data requirements. We then compare the performance of the proposed algorithm with the one of a centralized approach.

Key-words: Power grids, Reinforcement Learning, Factorization, Mutual Information

Contents

1. Introduction

Power grids are complex infrastructures that allow the transportation of energy from production plants to end consumers, with the use of thousands of kilometers of powerlines. Transmission System Operators (TSOs) such as Reseau de Transport d'Electricitée (RTE) in France work to provide a constant supply of electricity, balancing the demand of the users with the offer of the production. In doing so, they must ensure that the power flowing through the transmission lines does not exceed an upper limit called thermal limit, that depends on the properties of the metals that compose the powerline. If too much power is forced and the thermal limit is exceeded for a long period, the materials can be damaged, fail in service and be disconnected. [8]. Operating a power grid involves routing power around the network in order to avoid overloads that may lead to blackouts, while minimizing the economic and energetic costs of the operations.
Nowadays, the operation is carried on by dispatchers, highly trained engineers that can perform several different actions, called *remedial actions* on the grid, such as:

- Operations on the lines
  - *Switching*: changing the interconnections between lines to redirect flows.
- Operations on generators
  - *Redispatching*: increase or decrease the amount of energy that fossil fuels generators inject in the grid;
  - *Curtail*: limiting the amount of energy that renewable generators inject in the grid.
- Operations on storages
  - *Storing*: modifying the amount of energy absorbed by the storages

The challenge is becoming increasingly complex as the energy production shifts from fossil fuels, which offer predictable and manageable output, to renewable sources, whose production is highly unpredictable and can only be reduced, leading to significant energy waste. The globalization of the energy market further complicates the issue as power grids expand and become more interconnected.
This increasing complexity motivated researchers to investigate whether Artificial Intelligence (AI) models could be used as assistants to dispatchers in operating power networks in the most efficient and secure way. AI a branch of computer science that focuses on developing algorithms capable performing tasks that typically require human cognitive functions. These models analyze large amounts of data to identify patterns and improve their performances. The more data they process, the more they are able to adapt to new and unseen scenarios. AI tools can serve as powerful assistants by making split-second decisions and providing human dispatchers with immediate remedial actions to tackle grid issues.
Since 2019, French TSO, RTE, has been organizing a series of online challenges called *Learning To Run a Power Network* (L2RPN) [8]. The goal of the challenges is to encourage the use of AI, and in particular Reinforcement Learning (RL) for the assistance of human dispatchers in addressing critical situations that may arise when operating power grids. The aim of these challenges has been to promote investigation into the network operation problem in a competitive context. For that purpose, RTE developed the open-source Grid2op simulator [4] to model and study a large class of power system-related problems and facilitate the development and evaluation of agents that act on power grids. The Grid2op simulator is a flexible tool, allowing researchers to accurately simulate power system dynamics for different networks while interacting with the environment through different types of actions.
Over the last few years, more and more funtionalities have been added to Grid2op. Starting from a small, oversimplified grid, the enviroments evolved, including storages and constraints that make the problem closer to reality. During the several editions of the challenge, various RL algorithms have been proposed to tackle the problem. In particular a branch of RL which makes use of Deep Neural Networks (Deep RL) has shown promising results. So far, centralized algorithms that consider the grids as a whole have been explores. However, As the grids grow larger the amount of samples and training time required by these centralized models becomes prohibitive. AI literature calls this phenomenon "curse of dimensionality". A different kind of approach, in which multiple agents address different parts of the problem in a distributed (and possibly independent) way, may help to mitigate the curse of dimensionality.
Power grid management is also one of the topics of interest of *AI4RealNet*, a project that receives funds from European Union's Horizon Europe Research and Innovation programme[1]. Its main goal is to address critical systems that can be modelled as networks (power grids, railway, and air traffic management). These systems are traditionally operated by humans, and the aim of the project is to develop AI solutions that can be used as assistants to complement and augment human abilities in dealing with increasingly complex problems.
This thesis takes place in the context of the partnership of Politecnico di Milano's partnership with AI4RealNet, and it fits into one of the areas of the project, sharing its final objective of optimizing power grid operations, by means of innovative RL approaches.

[1] `https://ai4realnet.eu/`.

1.1. Contributions

Our proposal is to decompose large MDP problems into sub-instances, by means of a clustering algorithm that makes use of Mutual Information as metric to discriminate whether two state or action variables are correlated or not. We group together the variables that are highly correlated to each other and define possibly independent sub-problems.
We define a Distributed Multi Agent RL scenario in which multiple agents independently tackle different sub-problems. Their partial solutions are then combined to solve the overall problem. While this approach sacrifices the Markovian Property and the assurance of finding an optimal solution, it significantly accelerates the learning process. As a result, agents need less time and data to arrive at a satisfactory solution.
In power grid operation, this translates in segmenting the networks into sub-grids that are operated by a dedicated agent.
The code used to collect the results we provide in this work is available on GitHub at: `https://github.com/ThatBerra/thesis_AI4realnet_distributed`

1.2. Thesis Structure

This thesis is structured as follows: Section 2 presents previous works that tackle similar problems, Section 3 provides the theoretical notions applied in our solution, in Section 4 a description of the problem is detailed, in Section 5 we propose our algorithm, alongside the associated pseudocode, Section 6 shows the results of our experiments and finally Section 7 concludes the work, proposing future research topics.

2. Related Works

RL for power grids. Most of the RL methods developed to control power grids are related to the L2RPN competitions. An overview of the solutions proposed during the first edition is reported in [12]. The winning solutions of the last two editions are presented in [5] and [2], respectively. All the solutions proposed so far severely restrict the action space and are based on a single agent acting on the entire grid.
Subsequent works have further explored RL algorithms beyond the scope of the competitions yet with the same limitations described above [3, 9]. On the other hand, [16] proposed for the first time a multi-agent RL approach as a solution to reduce the action space by creating an agent for each substation and specializing it on its own topological actions only. A rule-based method based on the line overloads decides which agent has to act. Similarly, [10] presents a hierarchy of agents with different combinations of high-level and low-level agents. However, these approaches still suffer two main limitations: (i) the observation space is not reduced as each agent observes the entire grid, (ii) the number of low-level agents is related to the number of configurable substations, thus requiring a large number of agents on grids of increasing size.
With our method, we would like to overcome such limitations by distributing the learning process across a smaller number of agents, each taking care of a specific sub-problem that has been identified in the data-driven decomposition of the original problem (for instance, controlling a zone of the power grid with several substations). Each agent has thus a reduced action space and, at the same time, can only observe the relevant state variables of its own sub-problem, resulting in a simpler learning problem.

Power grid segmentation. The segmentation of large-scale power grids into zones is crucial for human operators when controlling the grid in real-time. Power grids are usually segmented into static zones that are redefined every year to study the grid efficiently in real-time. Typically, the segmentation has been computed using analytical methods that have been more extensively explored in the field of power systems [11]. Only recently, a data-driven approach using machine learning has been proposed [13]. By simulating the effects of a specific intervention of human operators (i.e., line disconnections on the other lines), the authors were able to create an adjacency matrix of a directed graph of lines on which they executed a graph clustering algorithm. The resulting clusters were used to segment the power grid, showing interesting results in different realistic scenarios.
Our method has in principle a similar objective of power grid segmentation but it is designed to be more general, i.e., to segment the entire state and the action spaces with respect to any kind of intervention and independently from the physical configuration of the grid. Most notably, our method is meant to be domain-agnostic and widely applicable to any complex decision problem.

3. Theoretical Background

3.1. Mutual Information

The basic assumption of our method for problem decomposition is that there is some correlation among the random variables describing the problem. We will measure this correlation with a nonlinear measure called Mutual Information (MI). This metric was introduced in information theory by Shannon in 1948 and it is a non-parametric measure of relevance, able to capture linear and non-linear dependencies among two variables. The intuition behind MI is to quantify how much knowing one variable reduces the uncertainty about another variable [17].
Knowing the real distributions of the two variables, MI can be exactly computed as:

$$I(S'_i, X_j) := \mathbb{E}\left[\log\left(\frac{p(s'_i, x_j)}{p(s'_i)\,p(x_j)}\right)\right] \tag{1}$$

However, it is not usually the real distributions of the variables are known, so most of the times MI is estimated from data. A survey on the most used estimators for MI is presented in [17].

Mixed MI Estimator. A suited choice for our problem is the estimator proposed in [6], which implements a mechanism to discriminate whether the variable is continuous or discrete and deals with them in a different way. They estimate the Mutual Information between a pair of random variables X, Y by taking a datasets of realizations $\{X_i, Y_i\}_{i=1}^N$. For each point in the dataset, their algorithm computes the k-nearest neighbor distance from the other points. If this distance is zero, then the data point is in a discrete component, otherwise it could be either continuous or mixed. Precisely, the procedure is detailed in Algorithm 1

Algorithm 1 Mixed Random Variable Mutual Information Estimator [6]

In: $\{X_i, Y_i\}_{i=1}^N$, where $X_i \in \mathcal{X}$ and $Y_i \in \mathcal{Y}$
Parameter: $k \in \mathbb{Z}^+$ - number of nearest neighbors

for $i = 1$ *to* N **do**
  $\rho_{i,xy}$:= the k smallest distance among $[d_{i,j} := \max\{||X_j - X_i||, ||Y_j - Y_i||\}, j \neq i]$
  if $\rho_{i,xy} = 0$ **then**
    $\tilde{k}_i$:= number of samples such that $d_{i,j} = 0$
  else
    $\tilde{k}_i = k$
  end if
  $n_{x,i}$:= number of samples such that $||X_j - X_i|| \leq \rho_{i,xy}$
  $n_{y,i}$:= number of samples such that $||Y_j - Y_i|| \leq \rho_{i,xy}$
  $\epsilon_i := \psi(\tilde{k}_i + \log(N) - \log(n_{x,i} + 1) - \log(n_{y,i} + 1)$
end for
return $\hat{I}(X;Y) := \frac{1}{N}\sum_{i=1}^N \epsilon_i$

3.2. Clustering

Clustering [18] is the problem of dividing a dataset into natural groups, called clusters, such that points that are similar are grouped together, while points that are dissimilar are assigned to different clusters. It is an Unsupervised Learning approach, meaning that it does not need a training dataset to learn the parameters but it just looks for an internal structure of the provided data.
There are many ways to evaluate how good a clustering is but since we will deal with data organized in matrices, supposed that we have a ground-truth of the real clustering, we will use the Frobenius norm of the difference between the estimated clustering matrix and the ground-truth. Calling the former $\widehat{I}_\mathcal{G}$ and the latter $I_\mathcal{G}$, the metric is computed as:

$$\ell\left(I_\mathcal{G}, \widehat{I}_\mathcal{G}\right) = \left\|I_\mathcal{G} - \widehat{I}_\mathcal{G}\right\|_F^2 = \sum_{i,j}\left(I_\mathcal{G}[i,j] - \widehat{I}_\mathcal{G}[i,j]\right)^2. \tag{2}$$

3.3. Reinforcement Learning

Power grid operations can be easily modeled as a sequential decision problem. At any time, indeed, dispatchers operating a grid must ensure that energy demand requested by loads must be satisfied by the production of generators, while ensuring that the thermal limit of each power line is not exceeded. To do this, human dispatchers at any time monitor the state of the grid and take *remedial actions* if they are needed.
RL is the learning paradigm suited to handle decision processes and it involves figuring out the best actions to take in different situations to maximize a numerical reward. Instead of being told which actions are best in every situation, the learner must experiment and discover which actions lead to the highest rewards. The problem is formalized as the optimal control of Markov Decision Processes [15].

Markov Decision Processes. An MDP is a tuple:

$$\mathcal{M} \coloneqq \langle \mathcal{S}, \mathcal{A}, P, R, H \rangle$$

where

- $\mathcal{S}$ is the set of all possible states of the environment;
- $\mathcal{A}$ is the set of all actions that can be performed;
- $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is a stochastic function that models the transition from one state of the environment to another, as consequence of an action. More in details, $P(s'|s, a)$ is the probability of moving to state s' when performing action a in state s;
- $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, specifically $R(s, a)$ is the reward the agent gets when performing action a in state s;
- $H \in \mathbb{N}$ is the time horizon of the problem.

MDPs are characterized by the *Markov property*, which ensures that both the transition probability and the reward function only depend on the current state and the chosen action. This means that the history of transitions is not to be taken into account when estimating a new one.
RL agents at each time step sense the state of the environment and choose an action that affects it. Moreover, the agent must have one or multiple goals that depend on the state of the environment. The environment receives the action, modifies its state according to it and produces a numeric reward that represent this goal. Agents explore MDP learning from their experience which actions yield higher rewards, in other words, which actions lead towards states close to the goal.

Policies A *policy* is a function that maps at each $t \in [H]$, each state to a (possibly stochastic) action, formally $\boldsymbol{\pi} = (\pi_t(a|s))_{t\in[H]}$. The objective of the agent is to find a policy that maximises the expected sum of the rewards obtained from the environment over the time horizon H. The expected return of a policy is given by its *value functions* (v_π and q_π).

- **State-value function** $v_\pi(s)$: it denotes the expected return starting from state s and then following policy π:

$$v_\pi(s) := \mathbb{E}_\pi\Big[\sum_{k=0}^{H} \gamma^k R_{t+k+1} \Big| S_t = s, \pi\Big].$$

- **Action-value function** $q_\pi(s, a)$: it denotes the expected return starting from state s, taking action a and then following policy π

$$q_\pi(s, a) := \mathbb{E}_\pi\Big[\sum_{k=0}^{H} \gamma^k R_{t+k+1} \Big| S_t = s, A_t = a, \pi\Big].$$

The parameter γ, which takes values $0 \leq \gamma \leq 1$, is called *discount rate* and it defines how much the agent is interested in future rewards. When $\gamma = 0$ the agent is said to be "*myopic*", as it is concerned only about immediate rewards. With γ approaching 1, the agent is more interested in future rewards.

Optimality. Solving an RL means to find a policy with a high reward over the time horizon. When dealing with MDPs, it is possible to exactly define the concept of *optimal policy* π_*. We call *optimal state-value function* the highest state-value function $v_*(s) = \max_\pi v_\pi(s)$ and *optimal action-value function* the highest action-value function $q_*(s, a) = \max_\pi q_\pi(s, a)$
Any policy that achieves the optimal state-value function for every state $s \in S$ is called optimal policy. Even if the optimal functions are unique, there may be multiple policies that achieve them, and they are to be considered equivalent.

3.3.1 Deep RL

RL can be enhanced with Neural Networks, that are used to handle high-dimensional data, automate feature extraction, learn intricate patterns and scale on bigger and more complex environments. Deep RL algorithms make use of Deep Neural Networks to approximate policies and value functions, and they have proven to be quite powerful tool that are able to tackle complex tasks as robotic control, playing video games and operating power grids.
A simple, yet stable and efficient Deep RL algorithm is Proximal Policy Optimization (PPO) [14]. It applies an actor-critic approach, meaning that it is composed by two networks. The "actor" observes the environment and selects the actions to be played at any time, while the "critic" estimates the value function and gives a feedback to the actor on how good were its choices. PPO implements a mechanism called *clipping* of the objective function, which allows to update policies with the biggest step possible, while avoiding drastic updates that may cause a collapse of the learning process.

3.3.2 Multi-Agent RL (MARL)

RL also works in multi-agent scenarios, in which no more a single agent but a set of agents interact with the environment by means of policies to maximize a reward function over a time horizon. Agents select individual actions, that together form a *joint* action. The action modifies the state of the environment according to its transition function and then each agent receives an individual reward, together with an individual observation. There are many paradigms of MARL [7]. They can be classified based on how the training is carried on:

- *Independent learning*: other agents are considered as part of the environment.
- *Centralized Training with Decentralized Execution*: each agent has its own action space, and a local observation, but the training is done in a centralized way.
- *Fully centralized learning*: all agents act according to a unique centralized policy.
- *Fully decentralized learning*: each agent learns its own independent policy, without any coordination.

And based on how agents interact with each other:

- *Communication-based learning*: agents share information while learning, so that they are able to coordinate their actions.
- *Team learning*: agents learn to act as a team to reach a common goal.
- *Competitive learning*: often modeled as zero-sum games, in which agents learn by competing against each other.
- *Hybrid approaches*: they combine features from multiple paradigms to enhance their performances.

MARL challenges. MARL can be a powerful tool, but it also introduces some challenges in the process [1]:

- Non-stationarity: a crucial aspect of MARL is that, given the fact that agents act with changing policies, the problem becomes non-stationary. Agents may adapt to other agents policies in a potential loop which could lead to unstable learning. MARL algorithms must be able to handle this type of non-stationarity.
- Optimality of policies: if it is intuitive to define optimality for single-agent RL as the policy that achieves the maximum expected return, in multi-agents the notion of optimality must take into account other agents' policies.
- Credit assignment: along with determining which past action contributed to a received reward, MARL must consider also the problem of finding out which agent's action contributed to the reward.
- Scaling: an increasing number of agent acting in the environment determines an increasing number of possible actions, and may lead to exponential growth of possible action combinations.

Distributed RL We use the Multi-Agent RL paradigm to define a Distributed RL approach, in which the original problem is not multi agent itself, but it is decomposed so that its parts can be handled independently by a number of agents, whose joint solutions form a valid action of the original, non decomposed problem.

3.4. Power Grids

Power generation is shifting towards renewable energy production from solar panels or wind turbines. Compared to fossil fuel combustion these sources of energy have a reduced impact on the environment, as their CO2 emissions are much lower. However, they are highly unpredictable, because the amount of energy produced depends on meteorologic factors. A new class of power grid users is becoming increasingly diffused. They are called prosumers and they own both loads and generator, meaning that they can either inject power in the grid or demand power from it. Lastly, energy market is shifting towards a globalized and more interconnected European grid.

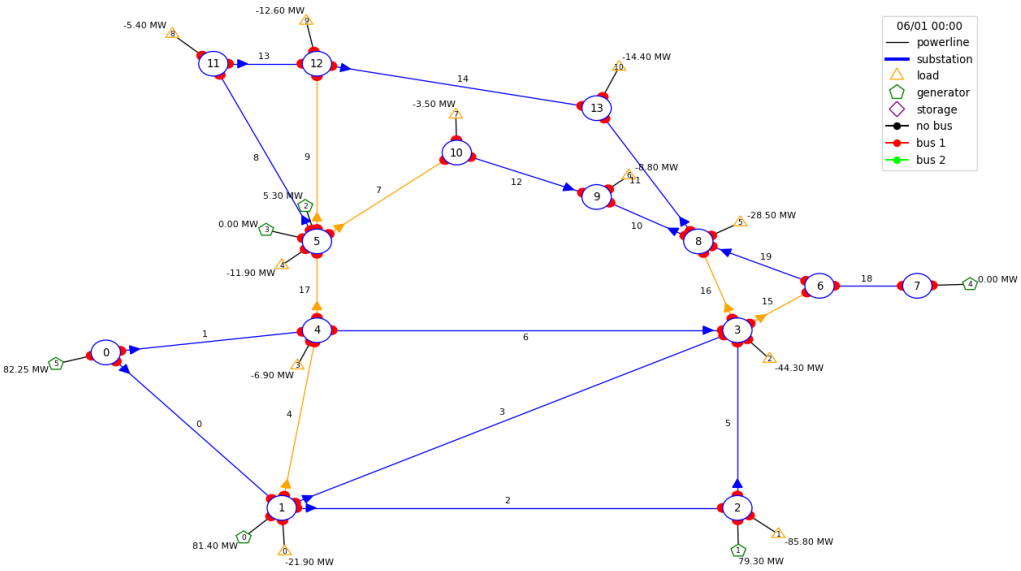


Figure 1: L2RPN case 14 power grid.

All these factor can be huge improvement in power grid efficiency and sustainability. However, they come to a price: from the point of view of dispatchers, operating a power network to balance energy demand and production, while keeping the grid safe becomes a very complex problem that requires coordinated decisions over time. To help dispatchers in taking decisions, RTE developed a simulated environment, called Grid2op, that casts the problem of power grid operations into a Markov Decision Process, so that it can be solved by means of Reinforcement Learning algorithms.

3.4.1 Grid Elements

Grids can be modeled as networks akin to the one in Figure 1. We can recognize several different elements, on which different types of action can be performed.

Generators. The green pentagons are *generators*. They can be either fossil fuels generators or renewable generators. On the former class dispatcher can perform a redispatching action, meaning that they can rise or lower the amount of power produced by the generator. Such actions are costly as they must take into account the economic cost of energy production. The only action that can be performed on renewable generators is curtailment, that means reducing the setpoint of the generator, so that just a given percentage of the total amount of power produced is injected in the grid. Since it is not possible to reduce the power produced by a renewable generator such as a solar panel, because it depends on the radiation of the sun, to avoid lines to be overloaded by excess power injected into the grid when it is not needed, the only way to operate on a renewable generator is to reduce the power injected into the grid. By doing so, however, some energy is produced but not used, and this implies a waste of energy.

Loads. The orange triangles are *loads*, representing the demand of the energy market. There are no actions that can be performed specifically on loads, their power levels must be treated as constraints.

Batteries. *Batteries* allow to store excess power when it is needed, acting as loads absorbing the overproduction. In alternative, if more power than the amount that generators can produce is demanded by the grid, batteries can act as generators, releasing some of the energy they stored to satisfy the grid's needs. Batteries adds flexibility to the network, but are quite complex to handle.

Substations. Loads, generators and storages are connected to *substations*, which are linked together by power lines. Substations can be seen as a set of wires, called buses. Each element of the grid can is connected to a substation on one of the buses. Connections of powerlines within substations define the grid topology, namely which elements of the grid are interconnected with each other. Topology actions can be performed on substation, by setting elements such as loads, generators and storages, or power lines on either one of the buses. This can be useful when a line is particularly overloaded, as it can create a node split as shown in Figure 2.

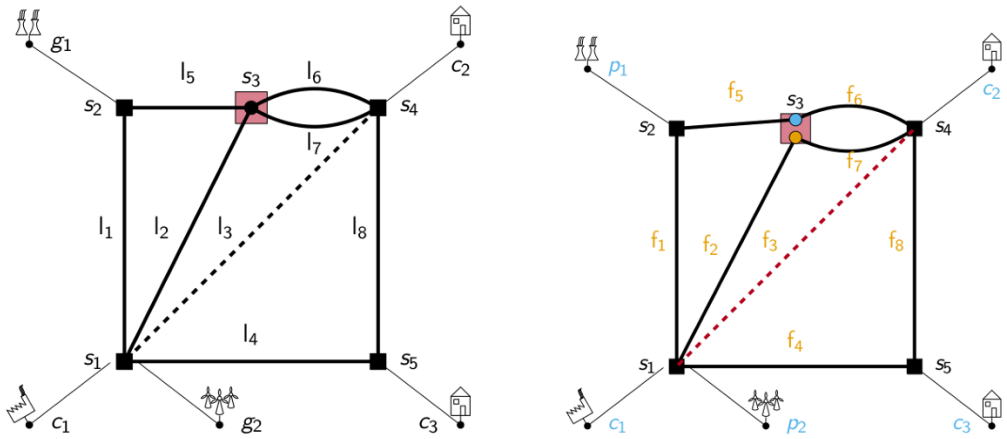


Figure 2: A simplistic network on the left, showing a simple network state, with no direct link between S1 and S4 since the dotted line indicates the line is switched out. The figure on the right shows a splitting action at substation node S3 which now creates an indirect link between node S1 and S4.[8]

These actions have no cost because they are just re-routing power flows inside the grid. However no storage or load can be isolated, so there are some substation configuration that must be avoided.

Power Lines. *Power lines* connect the substation, carrying power flows between elements of the grid. They are characterized by a thermal limit that must not be exceeded, otherwise the line is automatically disconnected to avoid breakdowns. Action on lines are connection or disconnection.

4. Problem description

In this section we present the problem we are facing, describing it in its multiple components. First we propose an overview of the main elements of the simulator, then we detail the formulation of the MDP and the challenges behind the decomposition, casting it into power grids scenarios.

4.1. Simulator

Grid2op [4] models the power grid scenario as an MDP, defining an environment, described by a network such as the one in Figure 1. In particular:

- Substations are modeled as if they had just two buses, even if in reality substations can have tons of them.
- Line connection and disconnection actions are available. Developers suggest to avoid to disconnect lines at any time and to reconnect disconnected lines as soon as possible.
- The default configuration of each environment sees all elements connected together on the first bus of all substations.
- For each generator a range is defined to limit the magnitude of redispatching actions that can be performed on that generator.
- Batteries were introduced in later edition of the challenge and have constraints similar to the ones of the generators.

A complete observation of the environment is available at each time step. It is composed by, among other information, the power line capacities, the power level of generators and loads and the configuration of the substations.
A wide choice of actions is proposed, to model many different remedial actions human dispatchers can perform. Given the huge dimension of observation and action vectors, in defining our problem we will consider just a fraction of them.

4.2. MDP definition

We need to formulate our problem as an MDP $\mathcal{M}$, as described in Section 3.3. Grid2op module helps in this, modeling complete and detailed observations and actions for each environment.

State. We reduce our observation to the *capacities* of the power lines. Capacities ρ_i are described as the observed power flows through the line divided by the thermal limit. Calling n the number of lines in the grid, the state of the MDP is $\mathbf{s} = (\rho_1, \rho_2, \ldots, \rho_n)$, with the capacity of each line being a state variable s_i. The state space is $\mathcal{S} \subseteq \mathbb{R}^n$, however, we are interested in keeping each $\rho < 1$: when the capacity of a line exceeds 1, the line is overloaded and the environment will automatically disconnect it.

Action. Since actions on loads, generators and batteries are quite expensive in terms of money and energy waste, many works suggest to focus on topology actions on substations [2], [5]. Our action vector is composed by the substations of the grid $\mathbf{a} = (sub_1, sub_2, \ldots, sub_m)$, with m being the number of substations, and each substation being an action variable a_i. Each substation has a possible number of configuration, hence the action space of action variable a_i is a set $\mathcal{A}_i$ containing all valid topological changes on the substation. Not all configurations are valid as some of them can isolate loads or generators. The number of valid topology changes depends on the number of connections on the substation, in particular, substations with 3 or less connections have just two equivalent valid configurations, hence it is useless to define actions on them.

4.3. Problem Decomposition

As the energy market transitions to a more interconnected European grid, managing these expanding networks becomes increasingly complex. Specifically, research must address the *curse of dimensionality*, which refers to the challenge that algorithms require larger datasets and more time to find optimal solutions as the problem's scale increases. We suggest breaking down the problem into smaller, independent sub-problems. This approach sacrifices the Markovian property, meaning there's no assurance of finding an optimal solution. However, by allowing different agents to work on portions of the entire problem and then combining their solutions, a sub-optimal but valid solution can be achieved more quickly and with a smaller dataset.

4.3.1 MDP decomposition

Let us consider a MDP where states and actions are multidimensional, formally $\mathcal{S} \subseteq \mathbb{R}^n$ and $\mathcal{A} \subseteq \mathbb{R}^m$, with $n, m \in \mathbb{N}$. At each time step $t \in H$, the state vector will be $\mathbf{s}^t = (s_1^t, s_2^t, ..., s_n^t)$. We call each s_i^t state variable. In a similar way, the action vector at time step $t \in H$ is denoted as $\mathbf{a}^t = (a_1^t, a_2^t, ..., a_m^t)$ and each a_i^t is called action variable. We want to investigate whether the problem can be divided into (possibly independent) sub-problems. Each sub-problem is described by a subset of the state variables and a subset of the action variables. The original problem $\mathcal{M}$ would be decomposed in k problems:

$$\mathcal{M} = (M_j)_{j=1}^k,$$

Each one of them being defined as:

$$M_j = \left(\tilde{\mathcal{S}}_j, \tilde{\mathcal{A}}_j, P_j, H, r_j\right)$$

The global transition probability and the global policy will be decomposed into independent transition probabilities $P_j : \tilde{\mathcal{S}}_j \times \tilde{\mathcal{A}}_j \times \tilde{\mathcal{S}}_j \to [0, 1]$.
Once the problem is decomposed multiple agents are instantiated and trained, each one targeting a specific sub-problem. At each time step, each agent receives only a portion of the global observation from the environment and chooses an action from a reduced action set, according to its own sub-problem. The reward is split among sub-problems as well, so that the agents receive the portion of the reward relative to the action they chose. Dealing with reduced problem, each agent finds a good partial solution in less time and with less data.

4.3.2 Grid segmentation

On grids, this means finding a segmentation of the power network, such that each segment can be considered as a standalone power grid. An example is proposed in Figure 3.
In the picture we can recognize three areas, divided by the colors of the power lines. The segmentation of the grid is purely based on topology, we want to provide a domain-agnostic method that allows to decompose any type of MDP with data collected by interacting with the MDP itself. Once the problem is decomposed, we can solve it in a distributed way, having a dedicated agent dealing with each sub-problem.

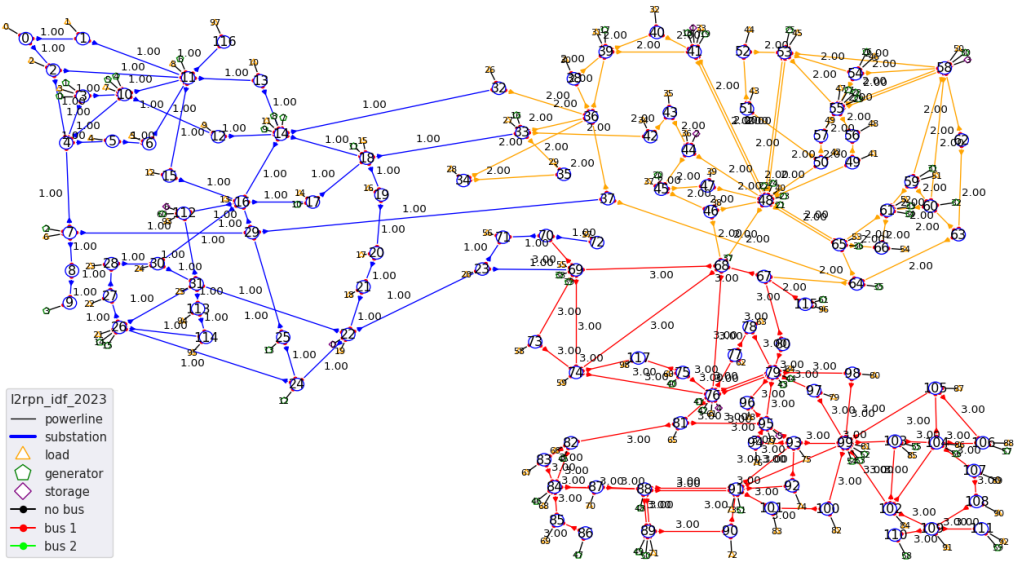


Figure 3: L2RPN 2023 segmentation.

5. Algorithm Proposal

5.1. Decomposition

We split the problem by means of a clustering algorithm. Clustering is a form of unsupervised learning, which given a dataset aims at dividing it into groups called clusters such that elements that are similar to each other are grouped together. A key factor in clustering is choosing an appropriate similarity or distance function to discriminate which elements are to be in the same cluster.

5.1.1 Variable Correlation

In the case of a RL problem as described before, we want to group together state and action variables that strongly influence each other over time, while keeping separated variables that have lower impact on each other. Ideally, if a problem can be perfectly decomposed, one should be able to find a clustering such that variables that are not grouped together have no impact on each other. Given a database of state action transitions on $\mathcal{M}$, called trajectory, collected with a sufficiently explorative policy, i.e. a policy that tries random actions with equal probabilities,

$$\tau = \left(\mathbf{s}^{t_0}, \mathbf{a}^{t_0}, \mathbf{s}^{t_1}, \mathbf{a}^{t_1}, \mathbf{s}^{t_2}, \mathbf{a}^{t_2}, \ldots, \mathbf{s}^{t_H}\right),$$

We want to measure how much each component of the state $\mathbf{s}^t$ and the action $\mathbf{a}^t$ vectors, is important in defining the values assumed by the components of the state vector $\mathbf{s}^{t+1}$ at later time steps. To do so, we rearrange the collected data in a database that has the form:

$$\mathcal{D} = \left\{(\mathbf{s}, \mathbf{a}, \mathbf{s'})\right\}_{t=1}^{T},$$

We can divide the dataset into two parts. We call *input variables* the state and action variables at present time t:

$$\mathbf{X} = (S_1, S_2, \ldots, S_n, A_1, A_2, \ldots, A_m) \in \mathbb{R}^{n+m},$$

and *target variables* the state variables at the next time step $t+1$:

$$\mathbf{S'} = (S'_1, S'_2, \ldots, S'_n) \in \mathbb{R}^{n}$$

Each element of $\mathbf{X}$ and $\mathbf{S'}$ can be considered as a random variable with its own probability function. The goal is to compute the statistical influence between each pair of variables (X_i, S'_j). We quantify this influence by estimating the Mutual Information between the two variables from the data collected during the interactions with the MDP. The use of estimators for statistical measures allows to approximate the measure even when it cannot be exactly computed due to the fact that the exact distributions of the variables are not known. However, this usually introduces bias. We collect the estimates of the Mutual Information on a $n \times (n+m)$

Algorithm 2 MI estimation

In: n - number of state variables
In: m - number of action variables
In: δ - threshold on Mutual Information
Out: MI - matrix of pairwise mutual information
Out: bin_MI - binary matrix of correlation

Initialize $MI \in \mathbb{R}^{n \times (n+m)}$
Run explorative policy on environment, collect history data $\mathcal{D} = \left\{(\mathbf{s'}, \mathbf{s}, \mathbf{a})\right\}_{t=1}^{T}$
$\mathcal{S}' = \left\{(s'_1, s'_2, \ldots, s'_n)\right\}_{t=1}^{T}$
$\mathcal{X} = \left\{(s_1, s_2, \ldots, s_n, a_1, a_2, \ldots, a_m)\right\}_{t=1}^{T}$
for $i = 0$ *to* n **do**
  for $j = 0$ *to* $n + m$ **do**
    Estimate mutual information $\hat{mi}(s'_i, x_j)$
    Shuffle s'_i
    Estimate bias $\tilde{mi}(s'_i, x_j)$
    $MI[i, j] = \hat{mi}(s'_i, x_j) - \tilde{mi}(s'_i, x_j)$
    $bin\_MI[i, j] = 1 \; if \; MI[i, j] > \delta, \; 0 \; otherwise$
  end for
end for
return MI, bin_MI

matrix. On that matrix we apply a clustering algorithm to group together variables which are highly correlated with each other. The resulting clustering represents the problem decomposition.
The algorithm for data extraction and MI estimation is proposed in Algorithm 2.
The line capacities that we selected as state variables, assume continuous values $\rho_i \in \mathbb{R} \; \forall \; i = 1, \ldots, n$. On the other hand, the action variables are categorical, taking values in discrete sets depending on the corresponding substation. Since we are computing the Mutual Information on mixtures of continuous and discrete variables, we need an estimator that can deal with them both. We adopted for this work the estimator proposed by [6], that is described in details in Section 3.1
This estimator $\hat{I}$ computed on the collected data is likely to yield some bias, that we try to remove by computing the Mutual Information $\tilde{I}$ shuffling the target variable values in the history so that the new quantity measures the noise in the data, then we remove the bias by subtracting it from the estimated $\hat{I}$. The threshold δ is used to determine whether the variables are correlated, if the MI is higher than the threshold, or not, if the MI is lower.

5.1.2 Clustering

In a first version of the algorithm, the clustering of state and action variables is found by means of a depth first search called alternatively on the rows and the columns of the binary matrix. The indices are then rearranged so that the binary Mutual Information matrix becomes (pseudo) block-diagonal. The blocks in the resulting matrix are the clusters representing the sub-problems.
Once a general clustering is found, an iterative algorithm is called to try and expand or reduce the blocks. This considers one block at a time and explores the cells of the matrix adjacent to it, expanding it to include one more row or column according to a score that is computed on the number of ones outside the block. The result of this algorithm is the decomposed MDP.
To evaluate the goodness of our estimation, if a ground truth of the real clustering is available, we compute a metric which tries to give an approximation on the error on the performance of the algorithm. We selected the Frobenius norm, as detailed in Equation 2, of the difference between the estimated matrix and the ground truth.

5.2. Distributed RL

The clustering process divides state and action variables into groups, each one of which is in fact a sub-problem. In grids, this means that we segment the network by placing the substations in different clusters. Once we do that we can define a distributed agent for each sub-grid. Each agent acts independently from the others on its area of the grid and it implements a PPO algorithm to solve the problem.

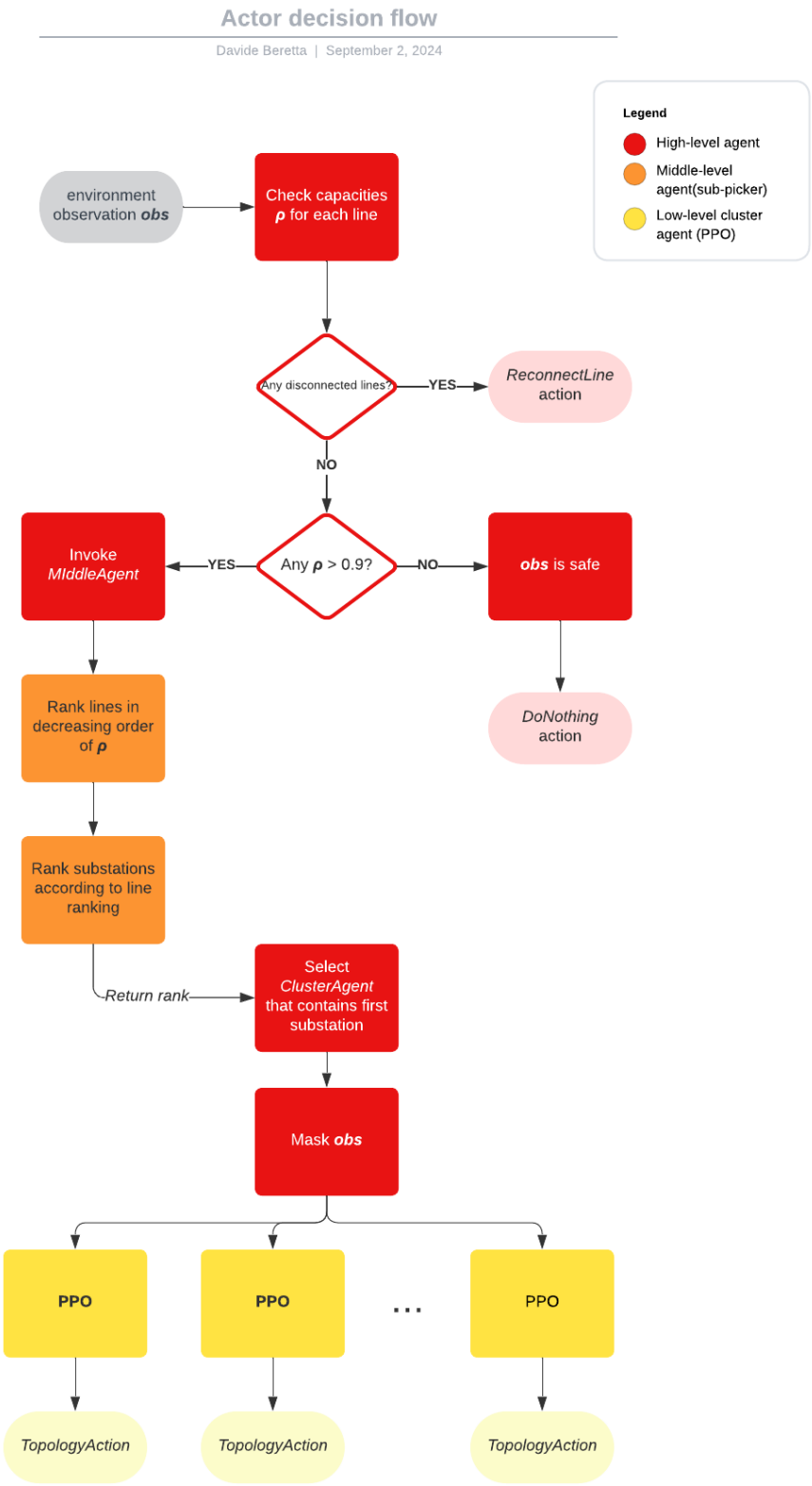


Figure 4: Decision flow of our actor

Since Grid2op does not provide multi-agent scenarios, and moreover it does not allow to perform more than one topological action per time step, we implemented an actor that is composed by a High-Level Agent, a Middle-Level substation picker Agent and a Low-Level PPO Agent for each sub-problem. The High-Level Agent coordinates the calls to the PPO agents and the interaction with the environment, using the structure proposed by [16] in their hierarchical approach. As shown by the diagram in Figure 4, The High-Level Agent decides if an action is needed, calls the Middle-Level Agent that selects the most crucial power line that needs intervention and calls the low level agent of the cluster the line belongs to. The low level agent receives a masked observation and based on its PPO network returns an action. During training, PPO agents are periodically updated to find an optimal solution to the sub problem.

Safe observations. An observation is considered *safe* if all power line capacities are below a given threshold, that we set at 0.9. As suggested by Grid2op developers and verified by various studies [12], it is good practice not to act if the grid is in a safe state, but take only *remedial actions* when the grid is at risk.

Disconnected lines. It may happen that a line is disconnected. This is caused by a power overflow in the previous step $\rho^{t-1} > 1$ or by the action of an adversary (introduced in "*l2rpn_wcci_2022*"). After a couple of timestep with any line disconnected the simulation terminates with a grid breakdown. To avoid this is mandatory that the agent tries to reconnect any disconnected line before attempting anything else.

Remedial action. If the grid is not safe but no lines are disconnected the agent must decide which lines, and hence which substations needs intervention. The middle-level agent ranks the lines according to their capacity values and then the high-level agent takes the substation with the highest number of critical incident lines. It retrieves the cluster the substation belongs to and calls the *act* function of the corresponding PPO agent. The agent has an internal mask that is applied to the observation, so that the agent only "sees" the power lines connected to the substations within its cluster.

6. Experiments

6.1. MDP Decomposition

Decomposing our problem $\mathcal{M}$ means to divide the grid in clusters, grouping together substations whose changes of topology have a good impact on the same power lines. We use Algorithm 2 to find an appropriate division.

6.1.1 Custom Environment

We tested the validity of our clustering algorithm on an MDP that can be perfectly decomposed. We defined a custom environment, with $n = 5$ state variables and $m = 3$ state variables. We divided the problem in the following clusters:

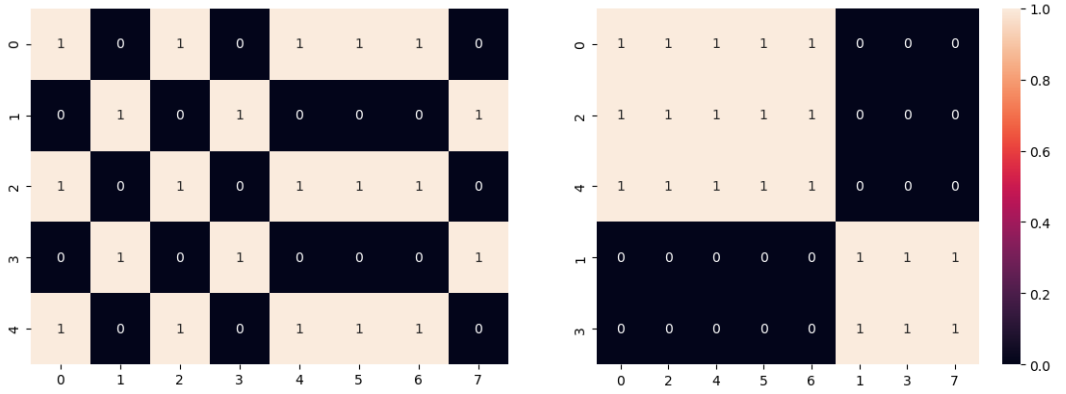
$$(1) : \widetilde{\mathcal{S}}_1 = (s_1, s_3, s_5),\ \widetilde{\mathcal{A}}_1 = (a_1, a_2)$$

$$(2) : \widetilde{\mathcal{S}}_2 = (s_2, s_4),\ \widetilde{\mathcal{A}}_2 = (a_3)$$

Given a state $\mathbf{s}$ and an action $\mathbf{a}$, each state variable at the next time step s'_i will assume a value extracted randomly from the ones of the variables in the same cluster. For instance, s'_1 will assume with the same probability the value of either one of $\{s_1, s_3, s_5, a_1, a_2\}$. Figure 5 shows the ground truth binary matrix (5a) and the pseudo block-diagonal one (5b).
State variables are initialized randomly as 0 or 1; at each iteration each action variable is randomly selected in the interval $[0, 1]$ to consider also a continuous scenario. Collecting $T = 10^5$ samples on this environment and setting the threshold δ as the 0.5^{th} quantile of each column, we were able to perfectly reconstruct the original decomposition of the problem. The Frobenius norm between the custom ground truth and our estimation, divided by the size of the matrices, is 0.02. This means that only one element out of 40 is different in our estimation. The estimated clustering matches the original one despite this different element. Figure 6 shows the results of the algorithm.

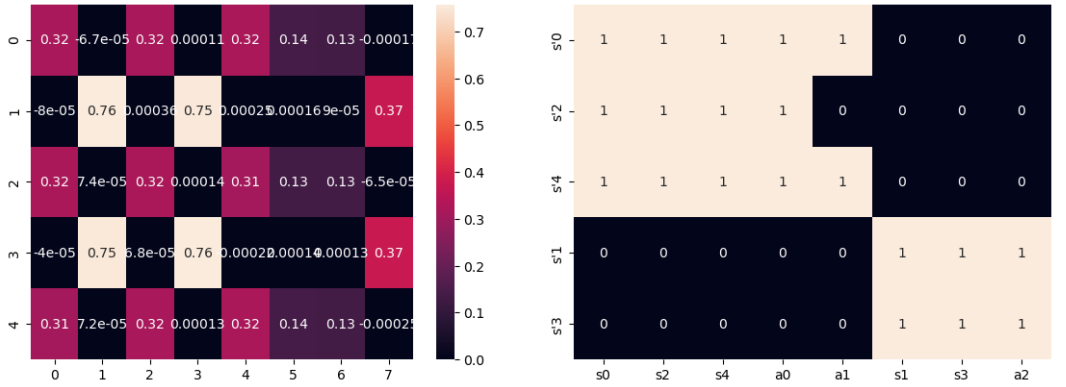
6.1.2 L2RPN Case 14

We run our algorithm on the simplest environment proposed by Gri2op, "*l2rpn_case14_sandbox*", which is described by the grid in Figure 1. We focused on the substations, and for each substation we collected a dataset by playing a policy that at each time step randomly selects a configuration among the available ones for that



(a) Custom environment relation matrix. (b) Custom environment pseudo block-diagonal.

Figure 5: Binary matrices of the custom environment



(a) Custom environment estimated MI. (b) Custom environment estimated pseudo block-diagonal.

Figure 6: Output of the clustering algorithm on custom environment

particular substation. We used these datasets to compute the columns of our mutual information matrix. We chose to apply a varying threshold, in particular we used the 0.7^{th} quantile of each column.
A ground truth of the division of this power grid is not available, however, we refer to the work of [13], in which a domain-expert analysis of the same grid is performed and a possible segmentation of the network in two areas is proposed. We explored different choices quantiles and we observed that he result with the 0.7^{th} quantile agrees with that segmentation.

$$(1) : \widetilde{\mathcal{S}}_1 = (line_0, line_1, line_2, line_3, line_4, line_5, line_6),$$
$$\widetilde{\mathcal{A}}_1 = (sub_1, sub_4, sub_2)$$

$$(2) : \widetilde{\mathcal{S}}_2 = (line_7, line_8, line_9, line_{11}, line_{12}, line_{13}, line_{14}, line_{15}, line_{16}, line_{19}, line_{20}),$$
$$\widetilde{\mathcal{A}}_2 = (sub_3, sub_5, sub_8, sub_{12})$$

Figure 7 shows how the segmentation appears on the "*l2rpn_case14_sandbox*" powergrid. In dividing the grid into areas we did not consider substations for which there are no available actions.
Figure 8 shows the raw Mutual Information matrix on the left (Figure 8a) and the matrix after it underwent the process of thresholding and block-diagonalization on the right (Figure 8b). We can recognize the two blocks highlighted on this binary matrix. The first is in the upper-left corner, the second in the lower-right one. The dimensionality of the matrix is reduced because all non-configurable substations were removed and some values of MI between lines and substations remain quite low. Since we are focusing in constructing an agent, in the following sections we will divide the matrix considering only the substations, each agent will see all the power lines connected to its substations.

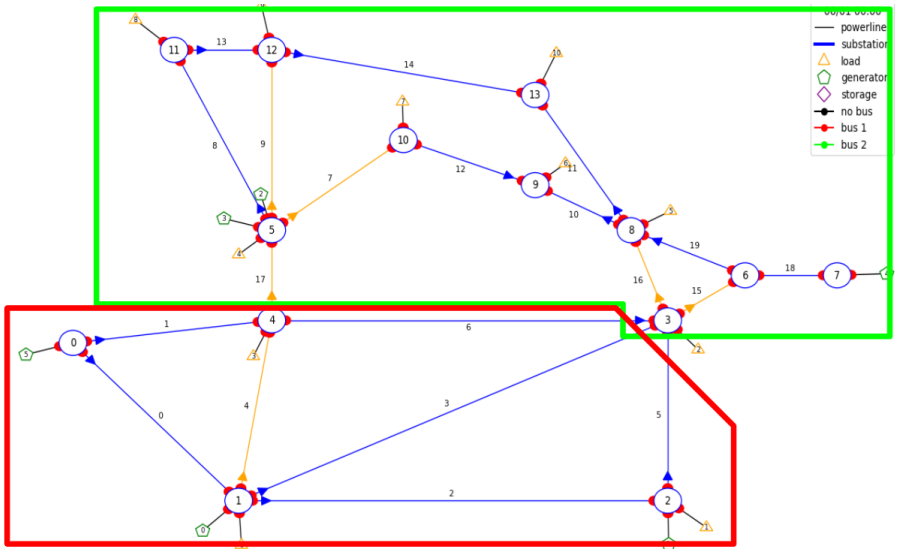
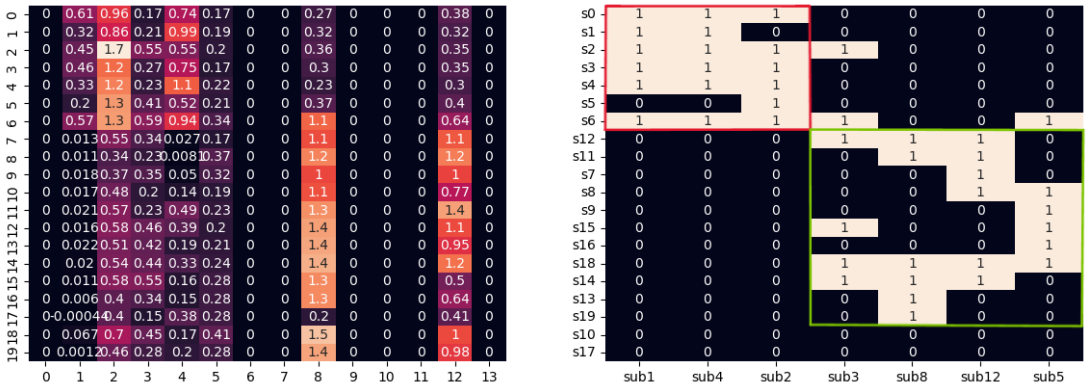


Figure 7: Case 14 optimal segmentation



(a) Case 14 Mutual Information matrix.

(b) Binary pseudo block-diagonal.

Figure 8: Estimated matrices

6.2. Distributed RL applied to power grid operations

6.2.1 Data preparation

Each Grid2op environment comes with a number of time series also called *chronics* in the documentation, that contain data to modify the input parameters of power flows between a given time step and the next one. They can be used to simulate the transitions during training and validation of agents.
The environment we chose, *L2RPN case 14*, provides 1004 time series, each one corresponding to a simulation of up to 8064 time steps. We kept 904 chronics for training and left the other 100 for validation. The training of the agents was carried on by iterating multiple times on the training dataset, in order to collect data for a higher number of iterations.

6.2.2 Reward and metrics

Reward. Grid2op provide a wide choice of rewards, able to capture different aspects of the power grid, alongside energy costs and waste. We chose the one called *CloseToOverflowReward*. It is strictly correlated to our actions and observations, as it depends on the capacities of the power line. Intuitively, it is higher when the grid is in a safe state, while it has its minimum when there are disconnected lines. In numbers:

$$CloseToOverflowReward(obs) = \begin{cases} 1 & \text{if } \rho_i < 0.9 \ \ \forall i = 1, \ldots, n\_lines \\ 0.5 & \text{if exactly one } \rho > 0.9 \\ 0 & \text{if more than one } \rho > 0.9 \end{cases} \tag{3}$$

We chose this particular reward because we want our agent to find good topology actions able to re-route power flows within the grid when a power line is overloaded, avoiding further congestion.

Performance metrics. We defined a performance to evaluate our models that computes the percentage of survived time steps over the total number of time steps provided. For a single time series, it is computed as:

$$r = \frac{n\_survived\_steps}{total\_steps} \tag{4}$$

Computing on an entire iteration on the training dataset, it becomes:

$$R = \frac{\sum_{i=1}^{n\_epochs} r_i}{n\_epochs} \tag{5}$$

,
where $total\_steps = 8064$ and $n\_epochs = 904$
The number of survived time steps is found by counting the time steps an agent is able to operate on the grid in the given time series, up until the grid goes to a breakdown and returns *done*. This number is always at least equal to the number of steps in which the environment is safe and it grows up to 1 if the agent is able to complete the simulation without a breakdown.

6.2.3 PPO parameter exploration

PPO comes with a set of parameters that can be modified to best suit the problem at hand. The first ones that we explored were buffer and batch size. The *buffer size* is the number of interactions with the environment the model collects before an update. Since most of the times our model does not act because the environment is in a safe state, in our case the buffer size is the number of topology action the agent performs before updating. The *batch size*, instead, is the number of training samples used in one update iteration. Each update is composed of 10 iterations and we selected four pairs of buffer size and batch size to compare. Figure 9 shows the performances for each pair during 100 iterations over the training set.
We can observe how the models with a smaller buffer size, the ones that are updated more frequently, early in the training reach a plateau, in which the performance does not improve, even though the model continues to be updated. We investigated this fact and looking into PPO *forward* method, we found that the vector of log probabilities from which the index of the action is sampled, at some points in the training contains an element with a value near 1 and all the other close to 0. This way the model stops the exploration of different actions and it always plays the ones it considers to be the best, even if it is sub-optimal, as we can observe from the orange line corresponding to the agent with $buffer\_size = 32$.
Increasing the buffer size, we can observe that the training seems to be much slower, as in the 100 iterations there is no significant improvement but a couple of peaks that are not exploited. However, with these hyperparameter choices, the models do not get stuck on a policy as the other two did.

6.2.4 Comparison with centralized approach

We selected the buffer and the batch size that obtained the best results during training and with the same setup we trained two other models, to compare the performance. The first implements a centralized approach, in which there is no division in cluster, and the only agent at each time observes the whole grid and can perform an action on any substation, the second implements a distributed approach, with the same scheme of our original model. In this version, Low-Level Agents receive the observation of the whole grid, and can act only on the subset of substations in their cluster. In the following we refer to this as *complete observations* model, as opposed to our *independent observations* proposal. Figure 10 shows the performances of the three models during training.
By looking at the plot, we can see how both the centralized approach and the one with the complete observations reach the plateau in the first iterations of the training and from there they stabilize on a much lower performance than the one of the model with independent observations. This is promising because it shows that our proposed model is able to learn how to operate on the grid, even in a distributed way, in which Low-Level Agents have a partial observation of the grid, and with this particular choice of parameters, it even seems to top the performance of a centralized approach that can observe the entire environment.

Learning Rate. To overcome the problem of the plateau we used two approaches. The first was to wrap PPO with an epsilon policy, that with probability ϵ selected a random action and with probability $1-\epsilon$ selected the output of PPO. We tried $\epsilon = 0.7$ and $\epsilon = 0.4$, but in both cases the approach was ineffective and the model did not learn.

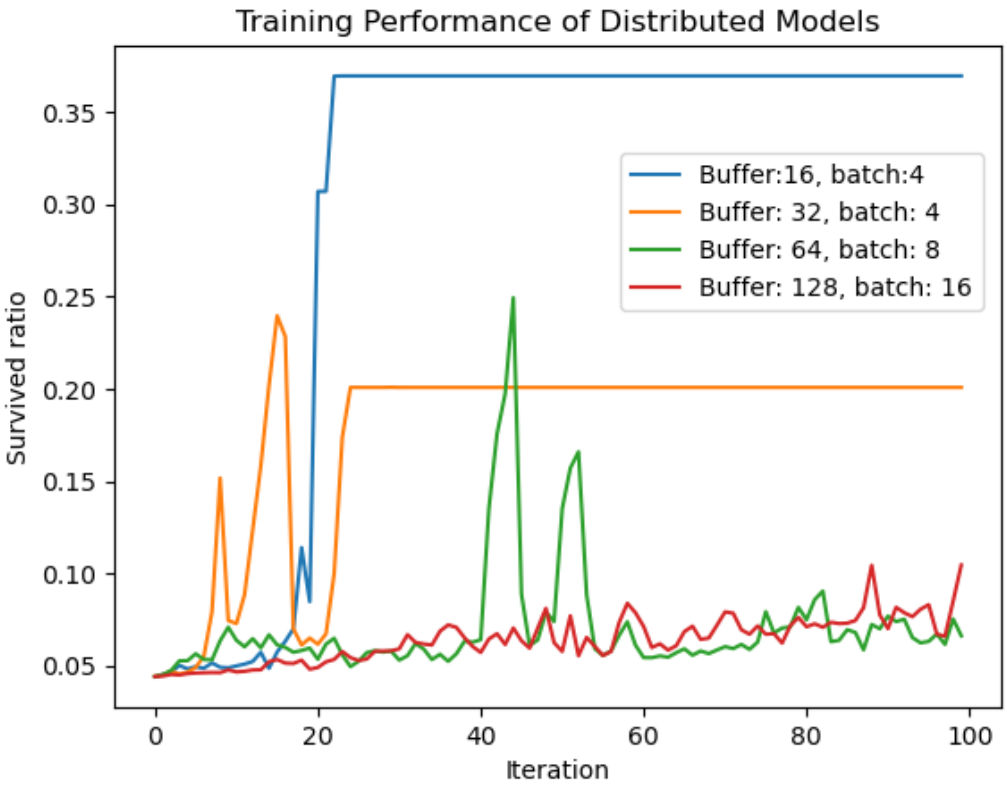


Figure 9: Comparison between different choices of *batch_size* and *buffer_size* on distributed models.

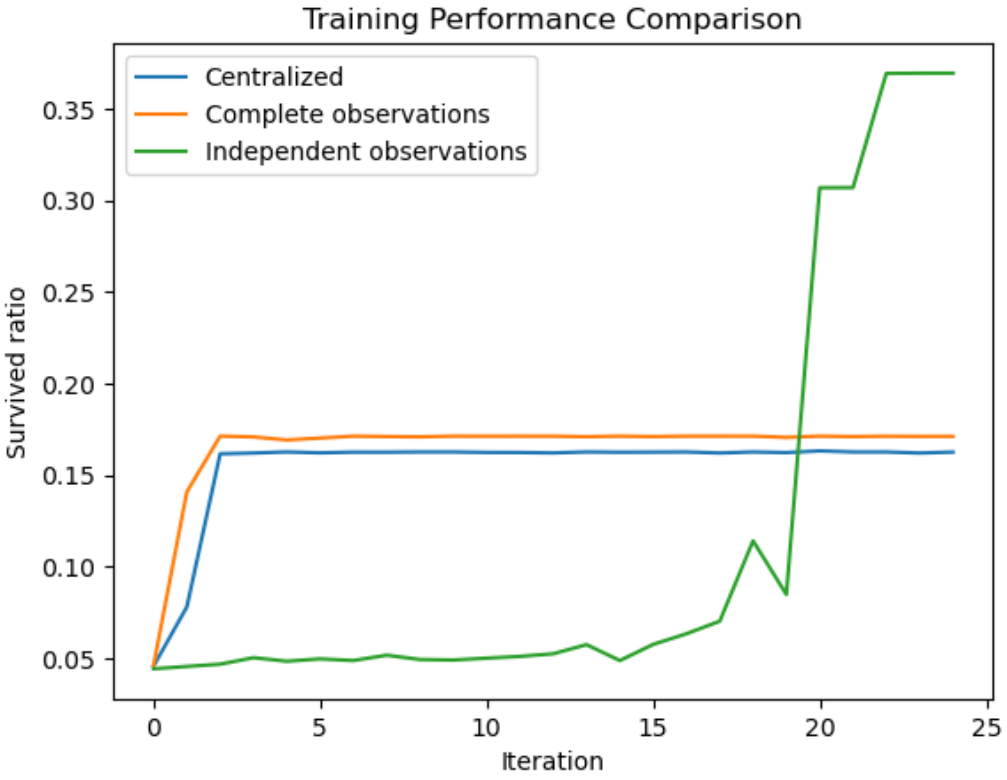


Figure 10: Comparison between centralized and distributed approaches.

In the second approach we modified the learning rate of PPO, reducing it from 3×10^{-4} to 3×10^{-5}. This choice seems to limit the plateau problem, as neither the centralized model nor the distributed one reaches it in the first 25 iterations. However, the best performance reached by the distributed algorithm is lower than the

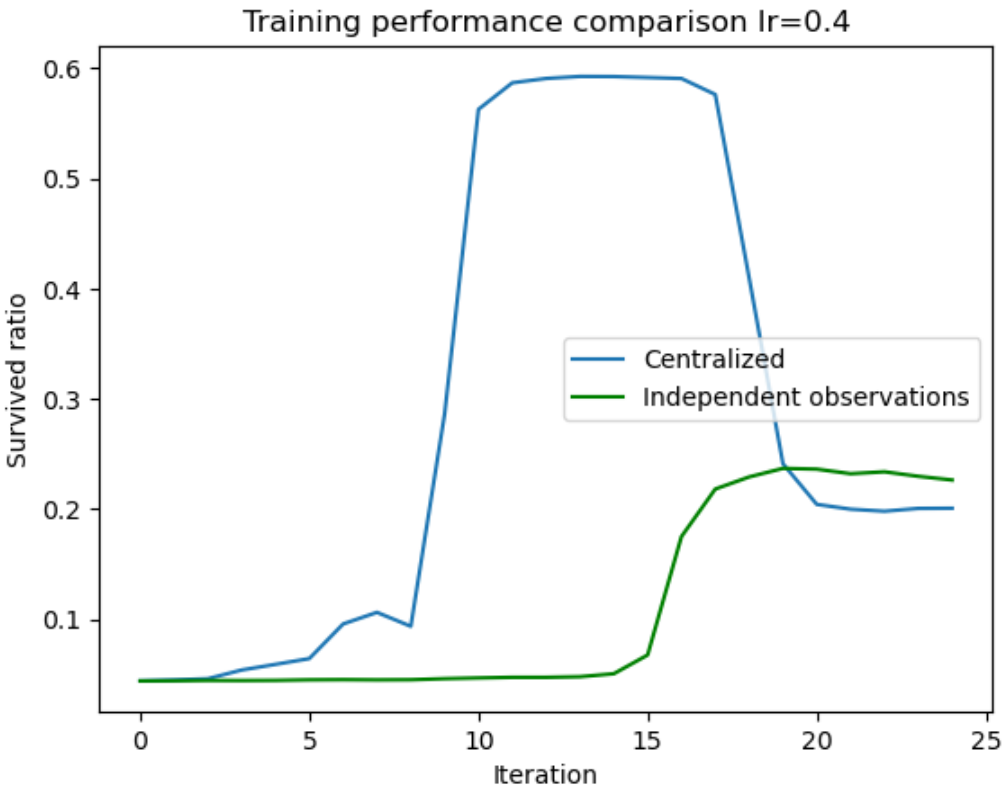


Figure 11: Comparison between centralized and distributed approaces with reduced learning rate.

plateau reached with an higher learning rate, while the centralized model achieves a much higher performance, even if it dorps at a certain point during the training. Figure 11 summarizes the performances of the two models over 25 iterations.

6.2.5 Validation

Using the validation dataset of 100 chronics that we defined in Section 6.2.1, we evaluated the performance of the best models we obtained on new unseen data. We selected the models that achieved the highest performances over the training set. In particular we loaded the centralized model trained with learning rate 3×10^{-5} from the 13^{th} iteration, in which it survived about 59% of the total time steps. As for the distributed model, we loaded the one with learning rate 3×10^{-4}, after it reached the plateau, in which it survived roughly $37 - 38\%$ of the time steps.
We run those two models on the validation dataset, computing for each chronic the metric as in Equation 4, and we plot the results on a barplot which shows three bars for each chronic. On the left in blue is shown the performance of the centralized model, on the right there is the one achieved by the distributed one, while the middle bar represent the performance of the *DoNothingAgent* provided by Grid2op, that do not act on the environment. The last 10 chronics are shown in Figure 12, The other ones are provided in the plots in Appendix A
The evaluation of the models on this independent dataset reflects the performances observed during training. The centralized model performs almost always better than the distributed one, which in some chronics is able to survive the same number of steps, while in other is much more similar to a *DoNothingAgent*.

7. Conclusions

This work proposes a distributed Reinforcement Learning paradigm to handle power grid operation problems. With this approach, a group of agents acts independently on a decomposed version of the original problem, to improve existing solutions for controlling power grids, while reducing time and sample complexity.
The division of the original problem into sub-problem is performed by means of a domain-agnostic algorithm for Markov Decision Process decomposition. Even without explicit knowledge of the problem, it is able to group the state and action variables in clusters based on the estimated values of mutual information between them, which

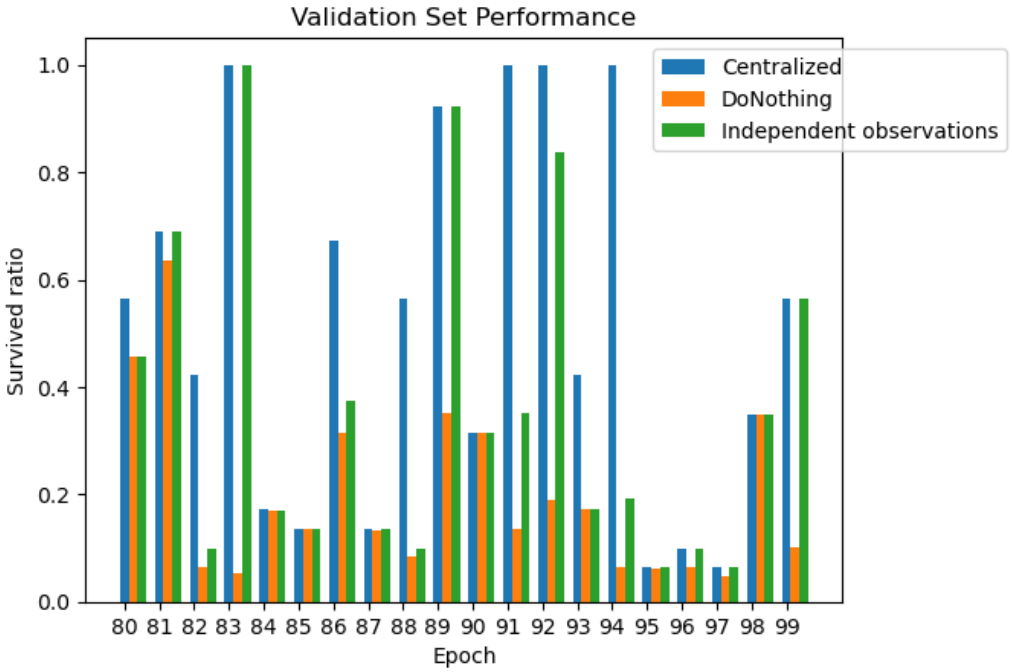


Figure 12: Model Evaluation. Chronics 80-100

provides theoretically grounded insights on how much pairs of variables are informative about each other. Our algorithm's results align with domain-expert analysis on a power grid benchmark obtained using an open-source simulator.
A distributed Reinforcement Learning that makes use of PPO as base learner is developed. An exploration of the parameters is performed to verify how performance and complexity varies depending on the different choices. The models are trained and tested in comparison with centralized approaches and show promising results, demonstrating potential applicability in power grid operation problems.

Future works Future works can explore correlation metrics different from mutual information, possibly reducing the number of samples required by its estimators. More robust clustering methods can be developed, so that variables can be grouped into blocks even in larger environments and matrices, while minimizing the number of non related variables within groups and related ones outside of them. A theoretical analysis is also required to understand how the choice of the policy for data collection influences the data distribution and how the choice of the threshold for building the adjacency matrix can be automatized.
Regarding distributed Reinforcement Learning, future research could investigate how algorithms different from PPO perform in this scenario and whether some form of communication or collaboration among agents can be applied to handle the fact that sub-problems are not independent.

References

[1] Stefano V. Albrecht, Filippos Christianos, and Lukas Schäfer. *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches*. The MIT Press, 2024.

[2] Artelys, 2023.

[3] Anandsingh Chauhan, Mayank Baranwal, and Ansuma Basumatary. Powrl: A reinforcement learning framework for robust management of power networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 14757–14764, 2023.

[4] B. Donnot. Grid2op- A testbed platform to model sequential decision making in power systems. . `https://GitHub.com/rte-france/grid2op`, 2020.

[5] Matthias Dorfer, Anton R Fuxjäger, Kristian Kozak, Patrick M Blies, and Marcel Wasserer. Power grid congestion management via topology optimization with alphazero. *arXiv preprint arXiv:2211.05612*, 2022.

[6] Weihao Gao, Sreeram Kannan, Sewoong Oh, and Pramod Viswanath. Estimating mutual information for discrete-continuous mixtures, 2018.

[7] Dom Huh and Prasant Mohapatra. Multi-agent reinforcement learning: A comprehensive survey, 2024.

[8] Adrian Kelly, Aidan O'Sullivan, Patrick de Mars, and Antoine Marot. Reinforcement learning for electricity network operation, 2020.

[9] Malte Lehna, Clara Holzhüter, Sven Tomforde, and Christoph Scholz. Hugo–highlighting unseen grid options: Combining deep reinforcement learning with a heuristic target topology approach. *arXiv preprint arXiv:2405.00629*, 2024.

[10] Blazej Manczak, Jan Viebahn, and Herke van Hoof. Hierarchical reinforcement learning for power network topology control. *arXiv preprint arXiv:2311.02129*, 2023.

[11] A Marot, B Donnot, S Tazi, and P Panciatici. Expert system for topological remedial action discovery in smart grids. In *Mediterranean Conference on Power Generation, Transmission, Distribution and Energy Conversion (MEDPOWER 2018)*, pages 1–6. IET, 2018.

[12] Antoine Marot, Benjamin Donnot, Gabriel Dulac-Arnold, Adrian Kelly, Aidan O'Sullivan, Jan Viebahn, Mariette Awad, Isabelle Guyon, Patrick Panciatici, and Camilo Romero. Learning to run a power network challenge: a retrospective analysis. In *NeurIPS 2020 Competition and Demonstration Track*, pages 112–132. PMLR, 2021.

[13] Antoine Marot, Sami Tazi, Benjamin Donnot, and Patrick Panciatici. Guided machine learning for power grid segmentation. In *2018 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, pages 1–6. IEEE, 2018.

[14] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[15] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 2018.

[16] Erica van der Sar, Alessandro Zocca, and Sandjai Bhulai. Multi-agent reinforcement learning for power grid topology optimization. *arXiv preprint arXiv:2310.02605*, 2023.

[17] Janett Walters-Williams and Yan Li. Estimation of mutual information: A survey. In Peng Wen, Yuefeng Li, Lech Polkowski, Yiyu Yao, Shusaku Tsumoto, and Guoyin Wang, editors, *Rough Sets and Knowledge Technology*, pages 389–396, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[18] Mohammed J. Zaki and Wagner JR Meira. *Data Mining and Machine Learning*. Cambridge University Press, 2020.

A. Appendix A

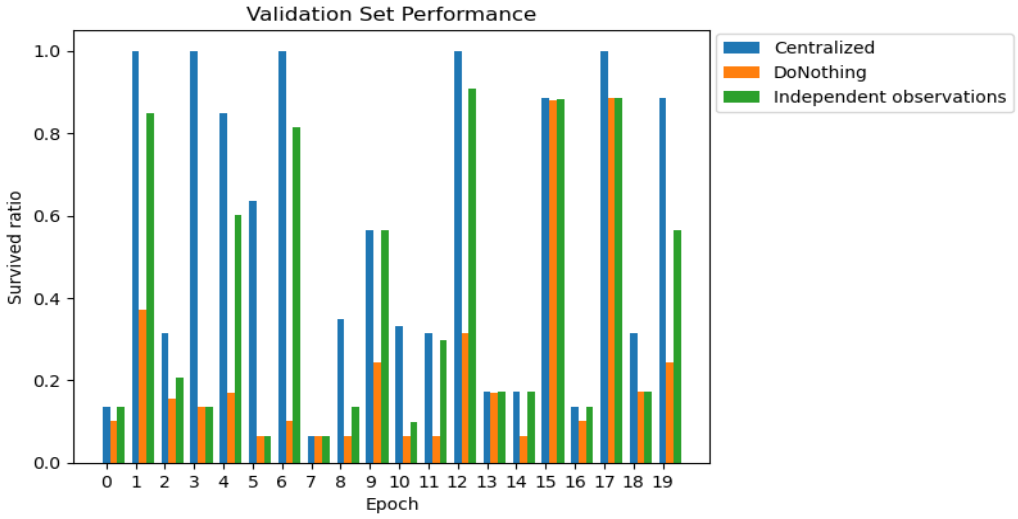


Figure 13: Model Evaluation. Chronics 1-19.

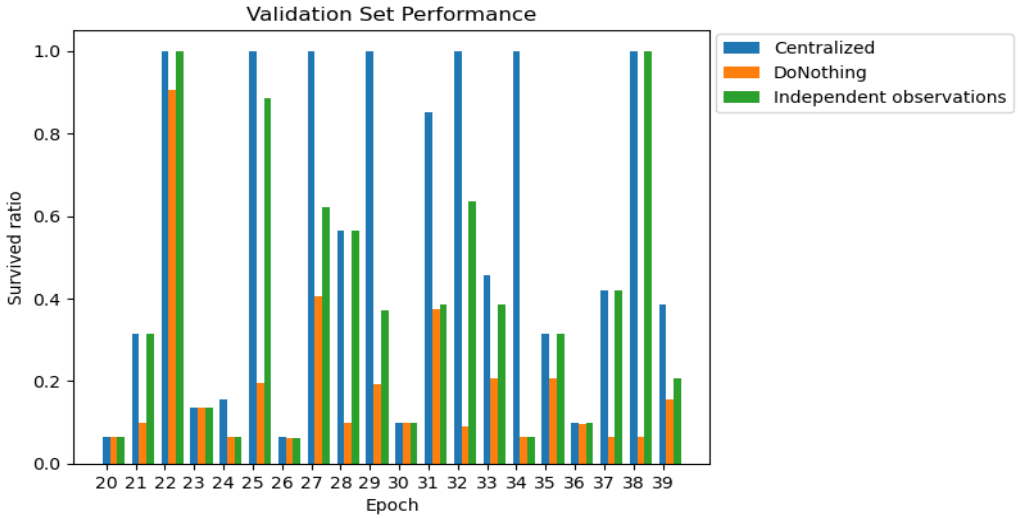


Figure 14: Model Evaluation. Chronics 20-39.

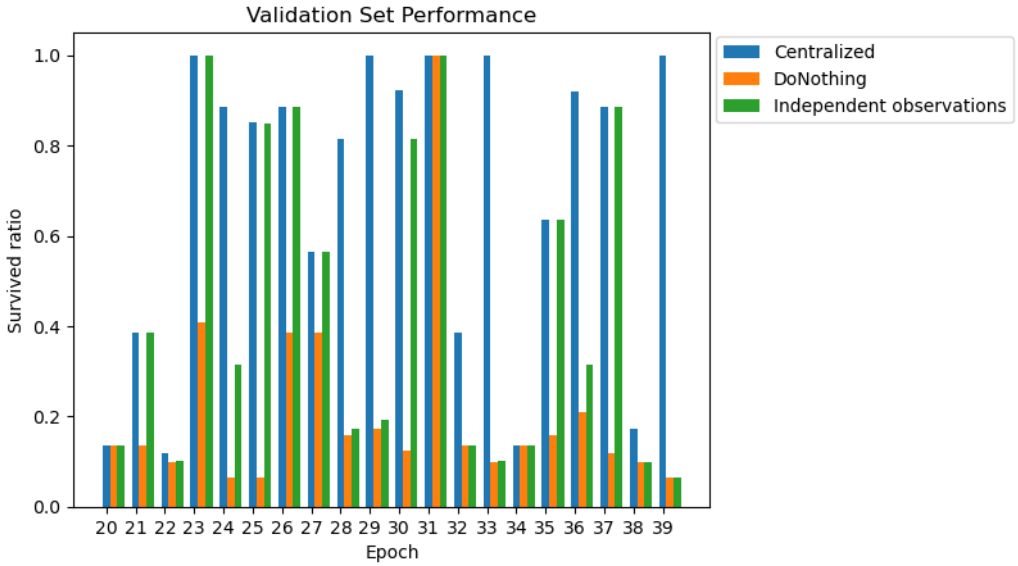


Figure 15: Model Evaluation. Chronics 40-59.

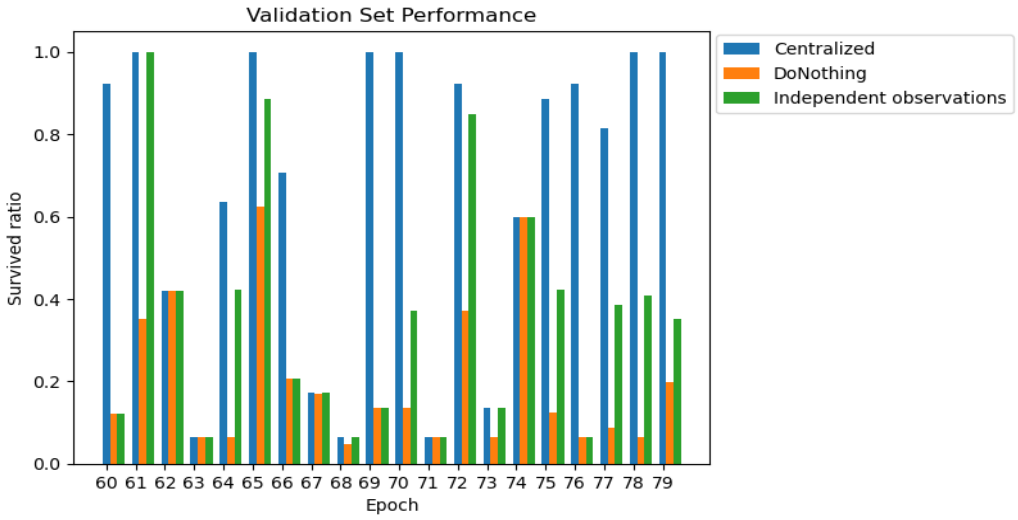


Figure 16: Model Evaluation. Chronics 60-79.

Abstract in lingua italiana

L'aumento delle rinnovabili nei mix energetici e la globalizzazione del mercato dell'energia rendono più complicato il problema del controllo di reti di distribuzione elettrica. Le recenti competizioni Learning To Run a Power Network hanno stimolato l'utilizzo di Reinforcement Learning come supporto per gli operatori umani nella gestione delle reti. Le soluzioni proposte finora prevedono dei limiti stringenti nello spazio delle azioni e utilizzano agenti in grado di osservare l'intera rete oppure di operare a livello della singola stazione. Nel nostro lavoro, proponiamo un approccio diverso, nel quale dapprima decomponiamo il problema tramite un algoritmo in grado di stimare direttamente dai dati la correlazione tra le componenti di stato e azione in modo indipendente dal dominio di applicazione. Coppie di variabili stato-azione molto correlate tra loro vengono raggruppate in modo da creare sottoproblemi più semplici e indipendenti. Su questa decomposizione definiamo un processo di apprendimento distribuito, nel quale ogni agente interagisce solamente con il suo sottoproblema, raggiungendo una soluzione parziale richiedendo meno tempo e dati. Alla fine, confrontiamo i risultati ottenuti da questo algoritmo con quelli di un approccio centralizzato.

Parole chiave: Reti di distribuzione elettrica, Reinforcement Learning, Algoritmi distribuiti, Mutua Informazione

Acknowledgements

Alla fine di questo lavoro voglio ringraziare per iscritto solamente due persone, dal momento che non posso condividere con loro il mio traguardo.

Papà, so che non leggerai mai le mie parole, ma sei stato la mia più grande fonte di ispirazione per la mia carriera universitaria. Molte volte mentre programmavo con il PC sulle gambe e la musica di sottofondo mi sei tornato in mente, nella stessa identica posizione, con le righe di codice che correvano veloci sul tuo schermo e le tue dita che ticchettavano sulla tastiera. In diverse occasioni mi sarebbe stato utile un supporto o semplicemente potermi sfogare con qualcuno che come te avesse competenze in quello che stavo facendo, ma nessuno in casa o tra i miei amici ne capisce molto, quindi ho dovuto risolvere da solo. Grazie papà, questa tesi è anche tua.
Eli, non so se è il momento giusto, ma in qualche modo spero che tu trovi l'occasione per leggere queste poche righe. Sei stata al mio fianco in quelli che considero gli anni più importanti della mia vita finora, nei quali ho definito che tipo di persona voglio diventare e sono certo che, qualunque sia il risultato, l'impronta del tuo passaggio sarà evidente. Di sicuro gli avvenimenti degli ultimi mesi hanno complicato un po' la situazione verso la fine, ma non voglio dimenticare tutto il sostegno che mi hai dato in questi anni, perchè la tua presenza è stata preziosa. Grazie, Eli, senza di te sarebbe stato tutto molto più difficile.