POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

Distributed Reinforcement Learning for Power Grid Operations

Laurea Magistrale in Computer Science and Engineering - Ingegneria Informatica

Author: Davide Beretta

Advisor: Prof. Marcello Restelli

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

Academic year: 2023-2024

1. Introduction

Power grids operations consist in providing a constant supply of electricity, balancing the demand of the users with the offer of the production. In doing so, operators must ensure that the power flowing through the transmission lines does not exceed an upper limit called thermal limit, otherwise blackouts may arise. The problem is challenging and it is becoming increasingly complex as the production shifts towards renewables. For this reason, French TSO, RTE, organized a series of challenges called Learning To Run a Power Network (L2RPN) to promote the application of Reinforcement Learning (RL) approaches to help power grid operators, called dispatchers, in the management of power grids. They developed an open-source simulator, called Grid2op [1], 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. While the winners of those competitions focus on centralized algorithms that tackle the problem as a whole, we propose a distributed approach, in which the problem is first decomposed in simpler sub-problems and then solved by multiple agents, one for each subproblem, with reduced time and sample complexity.

2. Problem Formulation

2.1. MDP definition

We formulate power grid operations problem as a Markov Decision Process (MDP). Since we are interested in keeping the power flowing through the power lines within a certain limit, we focus on the portion of the observation describing the capacities ρ of the lines, namely the ratio between the actual power flowing through the line and its thermal limit. Calling n the number of lines in the grid, the state of the MDP is $\mathbf{s} = (\rho_1, \rho_2, \ldots, \rho_n)$. As actions we only consider topology actions, as they are able to re-route power flows within the grid without any particular cost, while other types of actions, such as redispatching, are quite expensive in terms of money and energy waste. Substations are composed of a set of wires, called buses. The elements of the grid (generators, loads, batteries and lines) are connected to a substation on one of the buses. Topology actions consist in moving the elements from a bus to another and they must satisfy some constraints, so that not all substations can be modified and some of the possible configurations are not valid. The action vector of the MDP is composed by the configurable substations, and each element a_i has its own action space $\mathcal{A}_i$ containing all valid topo-

logical changes on the substation.

2.2. Problem decomposition

Given an MDP $\mathcal{M}$ defined over a time horizon H, at each time $t \in [0, H]$ the state vector is $\mathbf{s}^t = (s_1^t, s_2^t, \ldots, s_n^t)$ and the action vector is $\mathbf{a}^t = (a_1^t, a_2^t, \ldots, a_m^t)$. The goal is to decompose $\mathcal{M}$ into k independent sub-problems $M_j = (\tilde{\mathcal{S}}_j, \tilde{\mathcal{A}}_j, P_j, H, r_j)$. The global transition probability and the global policy will be decomposed as well into independent transition probabilities $P_j : \tilde{\mathcal{S}}_j \times \tilde{\mathcal{A}}_j \times \tilde{\mathcal{S}}_j \to [0, 1]$.
When dealing with power grids, this means finding a segmentation of the network, such that each area can be considered as a standalone power grid.

3. Algorithm Proposal

3.1. Decomposition

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. We run a sufficiently explorative policy, i.e. a policy that tries random actions with equal probabilities, on the MDP $\mathcal{M}$, and we collect a trajectory of state action transitions:

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

We can divide the obtained dataset into *input variables*, which are the components of the state and the action vector 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*, which are the state variables at next time step $t + 1$:

$$\mathbf{S}' = (S_1', S_2', \ldots, S_n') \in \mathbb{R}^n$$

Each element of $\mathbf{X}$ and $\mathbf{S}'$ is a random variable with its own probability function. We quantify the statistical influence between each pair (X_i, S_j') by means of a metric called Mutual Information, that measures the reduction in uncertainty that how much knowing one variable reduces the uncertainty about another variable [2]:

$$I(S_i', X_j) := \mathbb{E}\left[\log\left(\frac{p(s_i', x_j)}{p(s_i')\,p(x_j)}\right)\right] \quad (1)$$

Computing the exact value is not possible, as it requires the actual probability distributions to be known, hence we make use of an estimator, proposed by [2], which is able to deal with both continuous and discrete variables.
Since estimators of statistical measures are likely to yield bias, we try to approximate the errors by computing the Mutual Information on a dataset in which the realizations of the target variables are shuffled. Then we subtract this noise from the original estimates. We collect the obtained values in a matrix $n \times (n+m)$ in which each entry contains the MI between the target variable on the row and the input variable on the column. Given the matrix of the MIs, and a threshold δ, we define a binary matrix that contains 1, meaning that the variables are correlated, where the value of MI is greater than the threshold, and 0, meaning that the variables are uncorrelated, where the value of MI is lower than the threshold.

3.2. Clustering

In this first version of the algorithm, clustering is performed in place on the MI matrix. Alternating rows and columns, we perform a depth-first search on the binary matrix, rearranging the indices of rows and columns so that the matrix becomes pseudo block-diagonal. The blocks in the result are the clusters representing the sub-problems.
Automatic detection of blocks is done by sequential expansion/contractions of each block along the diagonal with the objective of minimizing the amount of 0s inside every block. The result of this algorithm is the decomposed MDP.
Provided that we have a ground-truth of the original clustering, we can assess the performance of this algorithm by computing the Frobenius norm of the difference between the obtained matrix and the ground-truth. Dividing the obtained value for the number of elements in the matrices, we obtain the error rate in reconstructing the original matrix.

3.3. Distributed RL

The clustering process divides state and action variables into groups, each one being in fact a sub-problem. We define a learning process in which each sub-problem is handled by a dedicated agent. In our grid scenario this means that

each agent acts independently on a subset of the configurable substations, observing a portion of the power lines of the grid.
Since Grid2op does not support multi-agent scenarios, 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 [4] in their hierarchical approach. Upon receiving an observation, the High-Level Agent verifies if there is any disconnected lines. If it is so, it returns an action to reconnect them. If all lines are connected, it checks whether the grid is safe, meaning that there are no lines with $\rho > 0.9$. If the state of the grid is safe, it returns an action that does nothing, otherwise it invokes the Middle-Agent, which returns a ranking of the substations according to the number of critical incidental lines. The High-Level Agent takes the most critical substation and invokes the *act* function of the Low-Level PPO Agent corresponding to the cluster the substation belongs to, providing a masked observation, so that the agent only observes its portion of the grid.

4. Experiments

The code used for the experiments is available on GitHub at: `https://github.com/ThatBerra/thesis_AI4realnet_distributed`

4.1. MDP Decomposition

Custom Environment. We tested the proposed algorithm on a custom MDP, designed to be perfectly decomposable. It has $n = 5$ state variables and $m = 3$ action variables. the first cluster is: $(1) : \widetilde{\mathcal{S}}_1 = (s_1, s_3, s_5)$, $\widetilde{\mathcal{A}}_1 = (a_1, a_2)$, while the second is: $(2) : \widetilde{\mathcal{S}}_2 = (s_2, s_4)$, $\widetilde{\mathcal{A}}_2 = (a_3)$. 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.
We colleced $T = 105$ samples on this environment and we set an adaptive threshold δ as the 0.5^{th} quantile of each column. With these parameters 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.

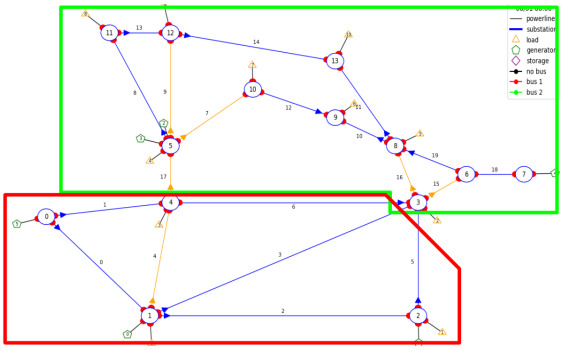

Figure 1: Case 14 optimal segmentation.

L2RPN case 14. We run our algorithm on the simplest environment provided by Grid2op, "*l2rpn_case14_sandbox*", which has $n = 20$ lines and $m = 14$ substations, of which just 7 are configurable. We focused on those substations and for each one of them we collected a dataset, running a policy that at each time step randomly selects a configuration among the available ones. With this data we were able to compute the columns of the MI matrix. We set a different threshold for each column, and we explored different choices, comparing the resulting segmentation with the one proposed by [3] in their domain-expert analysis of the grid. Using as threshold the 0.7^{th} quantile of each column we were able to obtain a segmentation that agrees with the one proposed in their paper.

4.2. Distributed RL applied to power grid operations

Given a dataset composed of 1004 time series, or chronics, each one corresponding to a simulation of up to 8064 time steps, we divided it in two part, using 904 chronics for training and the remaining 100 for validation. The reward selected for training the agents is called *CloseToOverflowReward* and it returns 1 if the grid is safe, 0.5 if there is one line is close to overflow, with $\rho > 0.9$ and zero if the number of lines close to overflow is greater than one. To visualize the performance of the models we defined a metric that computes the ratio between the number of time steps the agent survived in the simulation and the total number of time steps available

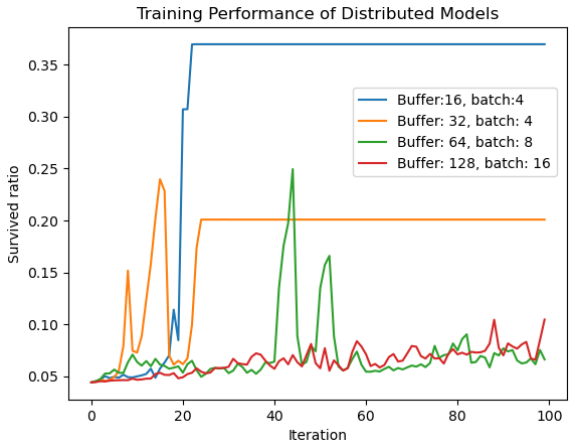

Figure 2: Comparison between different parameter choices.

(8064). In the case of training the value is computed for each iteration as an average over the number of chronics.

PPO parameters exploration. We explored different combination of buffer size, namely the number of interactions the agent collects before an update, and batch size, that is the number of samples considered in each update iteration. We observed that for low values of buffer size it happens that the training performance reaches a plateau in the early iterations and it does not improve with further update of the models. This does not happen for higher buffer size values, but the training is much slower and the performance does not improve much.

Comparison with centralized approach We selected the pair of buffer and batch size that achieved the highest performance in the training, i.e. $buffer\_size = 16$ and with those we trained a centralized model, in which a single agent can observe and act on the entire grid, and a complete observation model, in which the Low-Level Agents observe the complete grid but can act only on the substation in their cluster. We observed that both models suffer the same plateau problem, that presents itself earlier in the very first iterations. Moreover, they seem to stabilize on a much lower performance than the one of our model with independent observations. This shows that the algorithm is able to learn how to operate the grid and with this particular choice of parameters it even seems to top the performance of an algorithm able to observe the whole state of the environment.

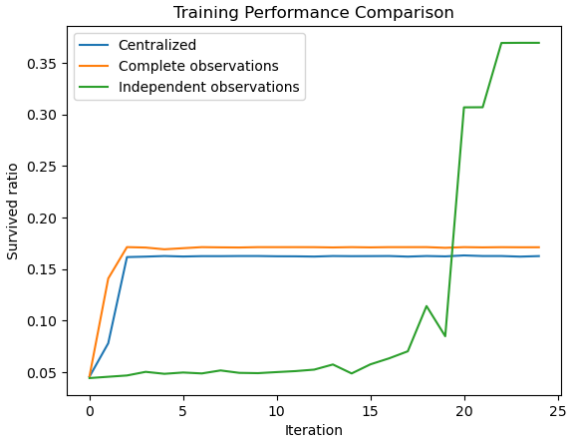

Figure 3: Comparison between centralized and ditributed approaches.

To address the problem of the plateau we modified the learning rate of PPO, reducing it from 3×10^{-4} to 3×10^{-5}. We trained a centralized and a distributed model with this setup and we compared the performances. Neither of the two reached a plateau, the centralized achieved a much higher best performance than the one obtained with the previous learning rate, even if it drops at a certain point during training. However, the performance of the distributed approach is lower than the previous one.

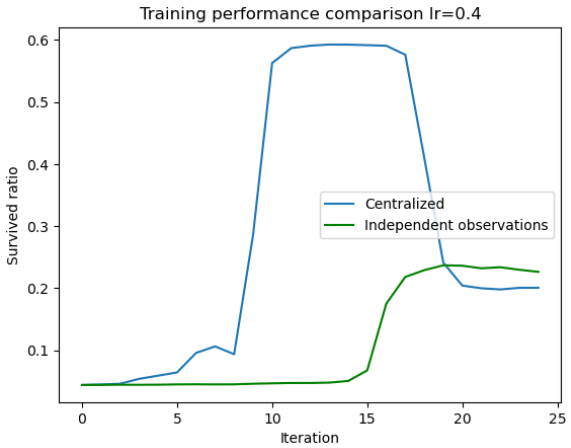

Figure 4: Comparison between centralized and ditributed approaches with a reduced learning rate.

Validation At last, we evaluated our best models on the validation chronics. We selected the centralized model with learning rate 3×10^{-5}

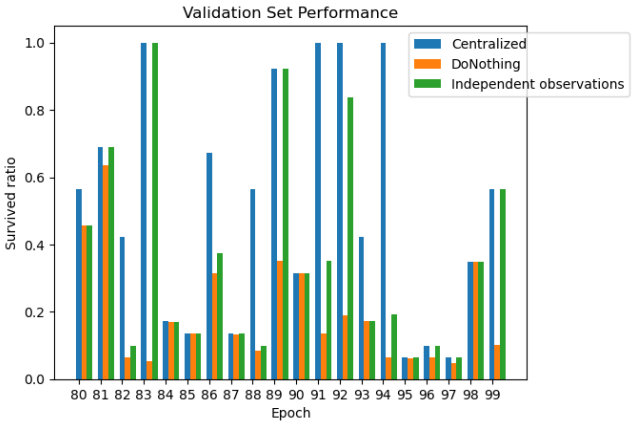

Figure 5: Model evaluation. Chronics 80-100.

at its 13^{th} training iteration, in which it achieved the highest performance of 0.59, and the distributed model with learning rate 3×10^{-4}, after it reached the plateau, which has a performance on the training dataset of 0.37. We computed the metric for each one of the chronics in the validation dataset and we plotted them in a barplot which has three bars for each chronic. On the left there is the value achieved by the centralized, on the right the one obtained by the distributed and in the middle the one achieved by an agent that does nothing at every time step. The results of the evaluation reflect the ones of the training, namely most of the time the centralized survives more time steps than the distributed.

5. Conclusions

In our work, we propose a distributed Reinforcement Learning paradigm to handle power grid operation problems. A group of agents acts independently on a decomposed version of the original problem. The decomposition is found by applying a domain-agnostic algorithm that clusters state and action variables of the corresponding Markov Decision Process using the estimated values of Mutual Information between variables. Once the decomposition is found, an agent that makes use of PPO as base learner is defined for each sub-problem. Different choices of parameters are explored and the models are then compared with centralized approaches, showing promising results and demonstrating potential applicability in power grid operation problems, which in turn is definitely better than the agent doing nothing.

References

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

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

[3] 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.

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