WINNING THE L2RPN CHALLENGE: POWER GRID MANAGEMENT VIA SEMI-MARKOV AFTERSTATE ACTOR-CRITIC

Anonymous authors
Paper under double-blind review

ABSTRACT

Safe and reliable electricity transmission in power grids is crucial for modern society. It is thus quite natural that there has been a growing interest in the automatic management of power grids, exemplified by the Learning to Run a Power Network Challenge (L2RPN), modeling the problem as a reinforcement learning (RL) task. However, it is highly challenging to manage a real-world scale power grid, mostly due to the massive scale of its state and action space. In this paper, we present an off-policy actor-critic approach that effectively tackles the unique challenges in power grid management by RL, adopting the hierarchical policy together with the afterstate representation. Our agent ranked first in the latest challenge (L2RPN WCCI 2020), being able to avoid disastrous situations while maintaining the highest level of operational efficiency in every test scenarios. This paper provides a formal description of the algorithmic aspect of our approach, as well as further experimental studies on diverse power grids.

1 INTRODUCTION

The power grid, an interconnected network for delivering electricity from producers to consumers, has become an essential component for modern society. For safe and reliable transmission of electricity, it is constantly monitored and managed by human experts in the control room. Therefore, there has been growing interest in automatically controlling and managing the power grid. Yet, an automatic control of a large-scale power grid is a challenging task since it requires complex yet reliable decision making. While traditional approaches have focused on controlling the generation or the load of electricity (Venkat et al., 2008; Zhao et al., 2014), managing the power grid through the topology control (changing the connection of power lines and bus assignments in substations) would be the ultimate goal. There are preliminary studies of the grid topology control in the power systems literature (Fisher et al., 2008; Khodaei & Shahidehpour, 2010), but due to its large, combinatorial, and non-linear nature, these methods do not provide practical solution to be deployed to the real-world.

On the other hand, deep Reinforcement Learning (RL) has shown significant progress in complex sequential decision-making tasks, such as Go (Silver et al., 2016) and arcade video games (Mnih et al., 2015), purely from data. RL is also perceived as a promising candidate to address the challenges of the power grid management (Zhang et al., 2020). In this regard, we present Semi-Markov Afterstate Actor-Critic (SMAAC), an RL algorithm that effectively tackles the challenges in power grid management.

One of the main challenges in RL for the real-world scale power grid management lies in its massive state and action space. We address the problem by adopting goal-conditioned hierarchical policy with the afterstate representation. First, we represent state-action pairs as afterstates (Sutton & Barto, 2018), the state *after* the agent has made its decision but *before* the environment has responded, to efficiently cover the large state-action space. The afterstate representation can be much more succinct than the state-action pair representation when there are multiple state-action pairs leading to an identical afterstate. Second, we extend this idea to a hierarchical framework, where the high-level policy produces a desirable topology under the current situation, and the low-level policy takes care of figuring out an appropriate sequence of primitive topology changes. Combined together, our

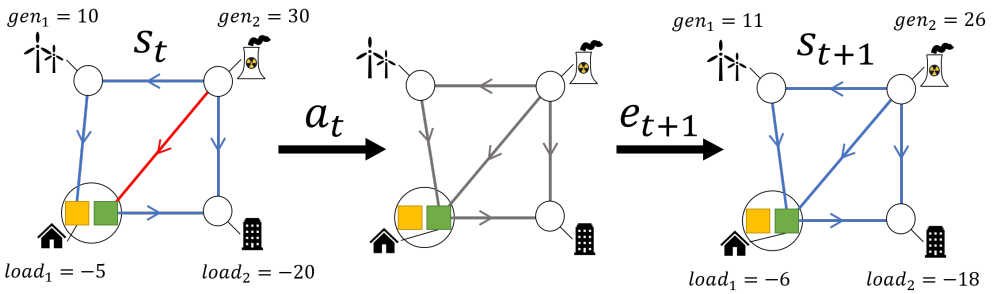

Figure 1: An example of a power grid with 4 substations, 2 generators, 2 loads, and 5 lines. Starting from the left, a bus assignment action a_t reconfigures the grid and then the next state s_{t+1} is determined by exogenous event e_{t+1}, such as the change of power demands in loads. The diagonal line was experiencing overflow, but the action a_t is shown to revert the overflow. The power loss also reduced from 15 to 13.

hierarchical policy architecture with afterstates facilitates effective exploration for good topology during training.

Our algorithm ranked first in the latest international competition on training RL agents to manage power grids, Learning To Run a Power Network (L2RPN) WCCI 2020. In this paper, we further evaluate our approach using Grid2Op, the open-source power grid simulation platform used in the competition, by training and testing the agent in 3 different sizes of power grids. We show that the agent significantly outperforms all of the baselines in all grids except for the small grid where the task was easy for all algorithms.

2 Background

2.1 Grid2Op Environment

We briefly overview Grid2Op, the open-source simulation platform for power grid operation used in the L2RPN WCCI 2020 challenge. The power grid is essentially a graph composed of nodes corresponding to *substations* that are connected to *loads*, *generators*, and *power lines*. The generator produces electricity, the load consumes electricity, and the power line transmits electricity between substations. The substation can be regarded as a router in the network, which determines where to transmit electricity. Grid2Op considers 2 conductors per substation, known as the double busbar system. This means that the elements connected to a substation, i.e. loads, generators and power lines, can be assigned to one of the two busbars, and the power travels only over the elements on the same busbar. Thus, each substation can be regarded as being split into two nodes.

The state of the power grid consists of various features such as a topology configuration (the connectivity of each power line and the bus assignment in each substation), as well as the amount of power provided by each generator, required by each load, transmitted in each line, and so on. The power supplied by generators and demanded by loads changes over time, and the power transmitted in lines also changes according to the current topology configuration together with supply and demand. In addition, each line has its own capacity to transmit electricity and can be automatically disconnected when there is an overflow of electricity.

The agent can apply actions on substations and lines to manage the power grid. The action on a substation, called *bus assignment*, assigns the elements in the substation to a busbar. The action on a line, called *line switch*, disconnects (both ends of line is assigned to neither bus) a line or reconnects a disconnected line. The agent is allowed to perform one line switch or one bus assignment action per step, and cannot successively perform actions on the same line or substation.

The power grid is simulated for a given period of time, typically for several days at a 5-minute interval. The simulation can terminate prematurely when the agent fails to manage the grid, i.e. (1) the amount of power required by loads are not delivered, which can happen if there are too many disconnected lines, or (2) a disconnected subgraph is formed as a result of applying an action. This is reflected into the failure penalty when measuring the performance of the agent, given by

the number of remaining simulation time steps upon termination. Another important performance metric is the power loss penalty, given by the amount of power disappeared during transmitting due to resistive loss. Thus, the goal of the agent is to operate the power grid both safely and efficiently by minimizing the failure penalty and the power loss penalty.

Figure 1 illustrates how the actions affect the state of the power grid using the bus assignment action as an example. The simulator provides 3 different sizes of power grids, (1) IEEE-5 is the power grid with 5 substations, (2) IEEE-14 is the power grid with 14 substations, and (3) L2RPN WCCI 2020 is the power grid with 36 substations. See Appendix A.1 for more details on the environment.

2.2 AFTERSTATES IN RL

Grid2Op provides a natural framework to use RL for operating power grids: we assume a Markov decision process (MDP) defined by $(S, \mathcal{A}, p, r, \gamma)$ to represent the RL task, where S is the state space, $\mathcal{A}$ is the action space, and $p(s_{t+1}|s_t, a_t)$ is the (unknown) state transition probability, $r_t = r(s_t, a_t) \in \mathbb{R}$ is the immediate reward, and $\gamma \in (0, 1)$ is the discount factor. We assume learning a stochastic policy $\pi(a_t|s_t)$, which is a probability distribution over actions conditioned on states. The state and action value functions under π are $V^\pi(s) = \mathbb{E}_\pi[\sum_{l\geq 0} \gamma^l r_{t+l}|s_t = s]$ and $Q^\pi(s, a) = \mathbb{E}_\pi[\sum_{l\geq 0} \gamma^l r_{t+l}|s_t = s, a_t = a]$ respectively.

As shown in Figure 1 in the previous section, the transitions in Grid2Op are comprised of two steps: the topological change that results directly from the action, and then the rest of the state changes that arise from exogenous events. This motivates the use of the *afterstate* (Sutton & Barto, 2018), also known as the post-decision state in Approximate Dynamic Programming (ADP) (Powell, 2007), which refers to the state *after* the agent has made its decision but *before* the arrival of new information.

Let us define the state S as $(\mathcal{T}, X)$ where $\mathcal{T}$ is the part of the state that deterministically changed by an action, and X as independent or affected indirectly from an action. Following the modeling in (Powell, 2007), the transition is decomposed into two parts using f^A and f^E:

$$s_{t+1} = [\tau_{t+1}, x_{t+1}] = f^E\left([\tau_{t+1}, x_t], e_{t+1}\right), \quad s_t^{a_t} = [\tau_{t+1}, x_t] = f^A\left([\tau_t, x_t], a_t\right), \tag{1}$$

where τ_{t+1}, the deterministic part of the s_{t+1}, is given by the the function $f^A(s_t, a_t)$, and x_{t+1}, the stochastic part, is given by the function $f^E(s_t^a, e_{t+1})$ where e_{t+1} is the source of the randomness in the transition sampled from some unknown distribution p^E. Note that e_{t+1} itself can be included as a part in x_{t+1}.

Using the afterstate has a number of advantages. For example, if the state and the action spaces are very large but the set of unique afterstates is relatively small, learning the value function of afterstates would be much more efficient. The value of an afterstate s^a under policy π is defined as $V^\pi(s^a) = \mathbb{E}_\pi[\sum_{l\geq 0} \gamma^l r_{t+l}|s^a = f^A(s_t, a_t)]$ and its recursive form can be written as :

$$V^\pi(s_t^{a_t}) = \mathbb{E}_{e_{t+1}\sim p^E, a_{t+1}\sim\pi}\left[r(s_t, a_t) + \gamma V^\pi(f^A\left(s_{t+1}, a_{t+1}\right))|s_{t+1} = f^E(s_t^{a_t}, e_{t+1})\right] \tag{2}$$

The optimal afterstate value function and the optimal policy can be obtained by iteratively alternating between the policy evaluation by Eq. (2) and policy improvement :

$$\pi_{new}(s_t) = \arg\max_{a_t}\left[V^{\pi_{old}}\left(f^A(s_t, a_t)\right)\right] \tag{3}$$

Note that we cannot gain much from the afterstate representation when using the individual power grid operations as actions since they result in unique changes in the grid topology. However, we shall see that the afterstate becomes very powerful when we consider the sequences of grid operations as the action space, where their permutations result in identical changes in the final topology.

3 APPROACH

We first present the state space, the action space and the reward function modeled in our approach. Then we briefly explain the unique challenge in Grid2Op and describe our approach to tackle the challenge. Finally, we will describe overall architecture of the RL agent.

3.1 Modeling States, Actions and Rewards

State We also define the state S in the Grid2Op environment as $(\mathcal{T}, X)$ where $\mathcal{T}$ is set of topology configuration (deterministically changed by an action) and X as various features as power demands and supplies (independent of the action), power being transmitted in each line (affected indirectly from the action) and so on. The detail about the features of states used in this work are provided in Appendix A.1.

Action We only consider bus assignment actions in our agent: we assume that it is desirable to have as many lines connected as possible since the overflow is less likely to occur when there are many routes for the power delivery. Thus, for line switch actions, we simply follow the rule always reconnecting the power lines whenever they get disconnected due to the overflow.

Let us define the number of substation as N_{sub} and elements in ith substation as $Sub(i)$. Each end of lines, generators, and loads in the substation can be assigned to one of two busbars, so the total number of actions is $|\mathcal{A}| = \sum_{i=0}^{N_{sub}} 2^{Sub(i)}$ (i.e. each action chooses one of the substations and perform a bus assignment therein). Following the approach taken by the winner of the previous challenge L2RPN 2019 (Lan et al., 2019), we made our agent act (i.e. intervene) only in hazardous situations. The condition for being hazardous is determined by the existence of a line of which the power flow is larger than the threshold hyperparameter. This naturally yields a semi-MDP setting for RL (Sutton et al., 1999).

Reward We define the reward in intermediate time steps to be the efficiency of the power grid, defined by the ratio of the total load to the total production, i.e. $\frac{load_t}{prod_t}$. Note that if the ratio becomes greater than 1, the episode terminates with a large penalty for the failure since the production does not meet the demand.

3.2 Actor-Critic Algorithm with Afterstates

The main challenge of Grid2Op environment is the large state and action spaces. For the power grid with 36 substations used in the L2RPN WCCI 2020 competition, there are about 70,000 actions that yields unique changes to the topology. We address this problem by adopting the actor-critic architecture, where the policy and the value function are represented by function approximators. In addition, we use the afterstate representation to capture many state-action pairs being led to to an identical afterstate by leveraging the transition structure, shown in Figure 1. For notational simplicity, all the derivations assume MDP in this section, which shall be extended to the semi-MDP setting in the next section.

We use function approximators for the afterstate value function $V_\psi(s_t^{a_t})$ and policy $\pi_\theta(a_t|s_t)$ parameterized by ψ and θ respectively. The actor is trained to maximize J_π and the critic to minimize L_V :

$$J_\pi(\theta) = \mathbb{E}_{s_t \sim D, a_t \sim \pi_\theta(\cdot|s_t)} \left[V_\psi(f^A(s_t, a_t)) \right] \tag{4}$$

$$L_V(\psi) = \mathbb{E}_{(s_t^{a_t}, s_{t+1}) \sim D} \left[\left(V_\psi(s_t^{a_t}) - r(s_t, a_t) - \gamma \mathbb{E}_{a_{t+1} \sim \pi_\theta(\cdot|s_{t+1})} \left[V_\psi(f^A(s_{t+1}, a_{t+1})) \right] \right)^2 \right] \tag{5}$$

where the replay buffer D stores the transition tuple $[s_t, s_t^{a_t}, r(s_t, a_t), s_{t+1}]$ for off-policy learning. The actor and the critic are trained using Soft Actor-Critic (SAC). Note that although the above equation defines a state-value critic, we can still train off-policy since it is essentially an action-value critic (i.e. an afterstate is defined by a state and an action).

In order to update the actor via reparameterization trick, the transition f^A must be differentiable, but it is not straightforward to define f^A, which maps from the bus assignment actions to the topology configurations, as a differentiable formula. In the next section, we will mitigate the problem by re-defining the action space.

3.3 Extension to goal-conditioned hierarchical framework

It is very challenging to take exploratory actions in the Grid2Op environment: if the agent takes random actions, the power grid would fail in a few time steps. For example, the agent with

the random policy would mostly fail in less than 10 time steps, whereas the agent with the no-op policy (naively maintaining the initial grid topology throughout time steps) would survive approximately 500 time steps on average. Thus, it is very difficult for the agent to explore diverse grid topology configurations that are significantly different from the initial ones, and thereby the random exploration policy (e.g. ϵ-greedy) would be often stuck at bad local optima that executes only one or two actions. Therefore, a more structured exploration is a key to successful training.

To this end, we extend the afterstate actor-critic algorithm to a two-level hierarchical decision model by defining the goal topology configuration as the high-level action. Specifically, we define the high-level actions as the goal topology configuration $g \in \{0,1\}^n$ where $n = \sum_{i=0}^{N_{sub}} Sub(i)$, which is learned by the high-level policy π^h. This leads to the temporally extended afterstate representation, given by $s_t^{g_t} = [\tau_{t+d} = g_t, x_t] = f^A([\tau_t, x_t], g_t)$ where t denotes the time a hazard occurs and d denotes the time interval next hazard occurs. Note that we can now take the full advantage of the afterstate representation, since the equivalence of many different sequences of primitive actions (i.e. individual bus assignment actions) that lead to the identical topology are now captured by the goal topology configuration.

In addition, exploration with goal topology is more effective than with primitive actions since the policy only needs to focus on *where to go*, i.e. the desirable topology under the current situation, without needing to care about *how to get there*, i.e. figuring out a suitable primitive action sequence that would yield the goal topology, with the help from an appropriate low-level policy. Finally, we can now use the reparameterization trick for the actor update in a straightforward manner since the result of f^A is merely a copy the action g_t.

The replay buffer D stores the transition tuple, $[s_t, g_t, r_{t:t+d}, s_{t+d}]$ where $r_{t:t+d} = \sum_{t'=t}^{t+d} \gamma^{t'-t} r_{t'}$. The high-level policy can be trained through the objective function of the actor and the critic written as:

$$J_\pi(\theta) = \mathbb{E}_{g_t \sim \pi_\theta^h} \left[V_\psi \left([g_t, x_t] \right) \right] \tag{6}$$

$$J_V(\psi) = \mathbb{E}_D \left[\left(V_\psi(s_t^{g_t}) - r_{t:t+d} - \gamma^d \mathbb{E}_{g_{t+d} \sim \pi_\theta^h} \left[V_\psi([g_{t+d}, x_{t+d}]) \right] \right)^2 \right] \tag{7}$$

As for the low-level policy, it is relatively simple to find the action sequence that changes the current topology into the goal topology: we just need to identify the set of substations that requires changes in the bus assignment and make appropriate reassignments therein. Thus, we take a rule-based approach for the low-level policy, $a_t = \pi_{rule}^l(s_t, g_t)$ where the $rule$ determines the order of substations to execute bus assignment actions. For example, we could impose a priority on substations such that the substations with the least room in the capacity make their bus reassignment first, because they are the ones requiring the most urgent interventions. In the experiments section, we compare the results using various rules including a learning-based approach.

3.4 Implementation

In order to leverage the interconnection structure of the power grid, we apply graph neural networks (GNN) (Scarselli et al., 2008). As illustrated in Figure 2, given the power grid with n substations, we reshape x_t in the state $s_t = [\tau_t, x_t]$, given as a flat vector in Grid2Op, into $(M, \tilde{x}_t)$, where $M \in \{0,1\}^{n \times n}$ is the adjacency matrix, and $\tilde{x}_t \in \mathbb{R}^{n \times k}$ is the node matrix with k features. We adopted the transformer (Vaswani et al., 2017) as the GNN block, where the adjacency matrix M is used for masking out the attention weights of nodes, following the transformer architecture proposed by Parisotto et al. (2020). The actor and the critic share the lower layers, consisting of GNN blocks and linear layers. Additionally, we add an entropy of policy to the objective function of the actor and the critic, following the SAC formulation. Details of the architecture are provided in Appendix A.2.

4 Related works

The topology control of the power grid through line switch has been previously studied in the power systems literature. Previous works, Fisher et al. (2008); Khodaei & Shahidehpour (2010), solve the optimal transmission switching problem by mixed-integer programming. From then, several

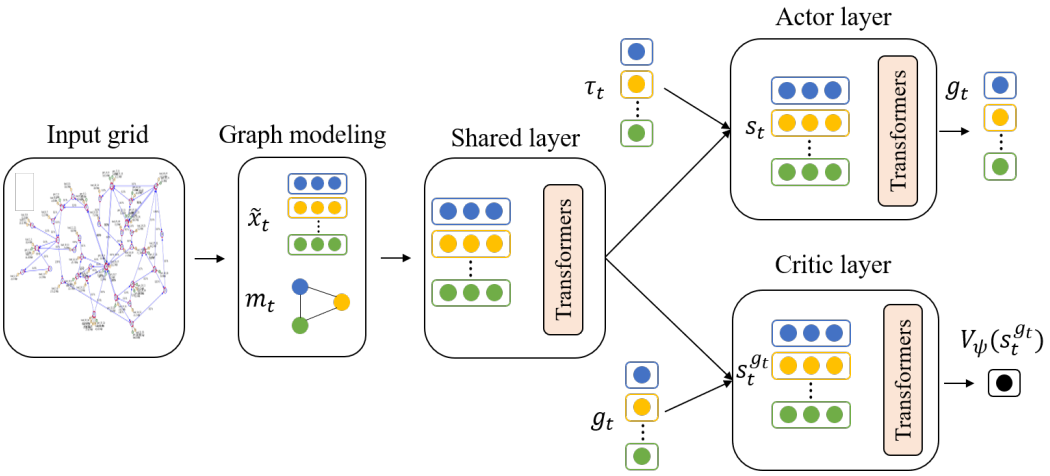

Figure 2: Overview of our model. The shared layer encodes x_t, the actor layer outputs the desirable topology g_t given the current state $s_t = [\tau_t, x_t]$, and the critic layer outputs the afterstate-value given the afterstate $s_t^{g_t} = [g_t, x_t]$.

Grid	N_{sub}	N_{line}	N_{gen}	N_{load}	$\lvert S\rvert$	$\lvert\mathcal{A}\rvert$	n
IEEE-5	5	8	2	3	74	58	21
IEEE-14	14	20	6	11	194	160	57
L2RPN WCCI 2020	36	59	22	37	590	66810	177

Table 1: Characteristics of the grids. N_{sub}, N_{line}, N_{gen}, N_{load} is the total number of substations, lines, generators, and loads. $|S|$ is the dimension of state, $|\mathcal{A}|$ is the number of unitary bus assignment actions, n is the dimension of topology configuration.

heuristics have been introduced to tackle the computational cost (Fuller et al., 2012; Dehghanian et al., 2015; Alhazmi et al., 2019). Recently, Marot et al. (2018) explores bus assignment, more complex than the line switch, and presents an algorithm based on expert knowledge, which shows the utility of bus assignment. Their algorithm can find remedial bus assignment action that can revert overflow with a high probability of success and acceptable computational time. Marot et al. (2020) models the power grid operations through line switch and bus assignment as a RL task and releases an opensource simulation for power grid operation in multi-step time horizons. Additionally, they held the international power grid management competition, L2RPN 2019 challenge.

The winner of the L2RPN 2019 challenge (Lan et al., 2019), where IEEE-14 is chosen for the competition environment, tackles the problem through supervised learning and guided exploration. They collect massive data sets from the simulator which can restore particular states, and pre-train an agent to generate a good initial policy. For exploration in the large action space, they use guided exploration instead of random exploration, where the agent simulates the top few actions with high action values before performs its action to the environment at every time step. They also design the agent to act only in hazardous situations, and they train it using dueling Deep Q-Networks (DQN) (Wang et al., 2016) and prioritized replay buffer (Schaul et al., 2016).

5 Experiments

5.1 Experimental Setup

Our experiments are conducted on the 3 power grids, IEEE-5 (smallest), IEEE-14, and L2RPN WCCI 2020 (largest, used in the challenge), provided by Grid2Op. Details of each grid are provided in Table 1. Each grid has a set of scenarios, and each scenario specifies the variations in the

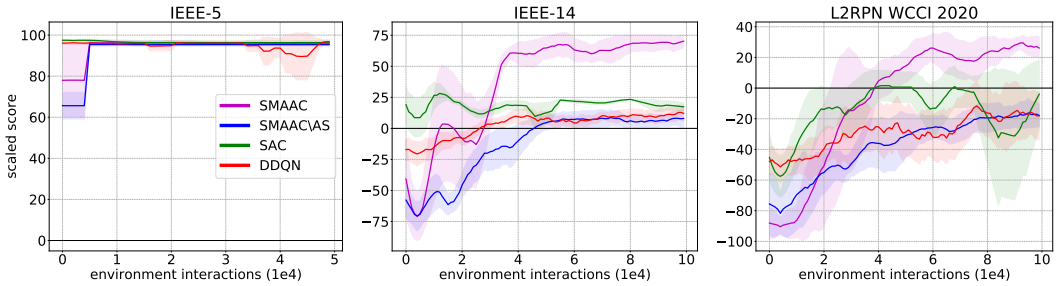

Figure 3: Training curves on 3 grids. Evaluation rollouts are performed every 1000 steps, and the shaded area represents the standard error.

	IEEE-5	IEEE-14	L2RPN WCCI 2020
SMAAC	98.18± 0.31	**69.66 ± 10.62**	**55.26 ± 5.82**
SMAAC\AS	93.79 ± 0.58	14.91 ± 14.08	11.58 ± 2.74
SAC	98.26 ± 0.06	43.93 ± 0.02	39.1 ± 2.94
DDQN	97.66 ± 1.04	29.11 ± 16.00	26.22 ± 7.39

Table 2: Performance on the 10 test scenarios averaged over 3 instances with standard error. Each the best policy is obtained from the one with the highest performance in the validation scenarios during training.

simulation such as the power supplies and demands at each time step. The length of each scenario is 864 time steps, which corresponds to 3 days at 5 minute time-resolution.

Since Grid2Op is relatively new to the research community, there are few RL methods applied to the grid topology control (to be more specific, there is no published work experimented on the largeest grid). Therefore we implement 3 baselines for performance comparison to verify the effectiveness of our method: (1) **DDQN** (Dueling DQN) has similar architecture as the last winner of the challenge, which learns the action-value function with the primitive action space (2) **SAC** is similar to DDQN but utilizes maximum entropy exploration following SAC algorithm. (3) **SMAAC\AS** is SMAAC without the afterstate representation, where we use *action*-value critic $Q^{\pi}(s, g)$. Thus, DDQN and SAC assume the MDP setting with primitive actions, SMAAC\AS assume the goal-conditioned SMDP setting but without the afterstate representation.

For fair comparison, all baselines encode the input state through the same GNN architecture, and the agents get activated only in hazardous situations. Details of implementation are provided in Appendix A.3.

5.2 Results

Figure 3 shows the total average scaled score of evaluation rollouts on the 10 validation scenario set during training: the scores are scaled in the range [-100,100], with the return of the no-op agent scaled and translated to 0. Each algorithm was trained and evaluated for 3 runs for averaging the scores.

As shown in Figure 3, all algorithms easily solve the smallest grid (IEEE-5). In the medium (IEEE-14) and the large (L2RPN WCCI 2020) grids, both DDQN and SAC perform poorly. DDQN performs slightly better than the no-op agent in the medium grid, and worse than the no-op agent in the largest grid. Exploring with primitive actions is extremely difficult since most of actions can lead to the disastrous termination, and thereby it cannot find grids other than the initial one. This yields the DDQN to be stuck at bad local optima, not much better than the no-op agent. SAC performs slightly better than DDQN in the larger grids. This is due to the sophisticated optimization scheme in SAC that is shown to have effect in a number of other RL benchmark tasks. However, in Grid2Op, the performance was barely better than the no-op agent due to the same challenge faced by DDQN.

	1 (ours)	2	3	4	5	6	7
Scaled score	75.72	66.21	48.62	26.60	17.98	4.31	0.07
CPU time (sec)	812.49	1406.45	1233.08	1322.02	116.43	96.56	118.58

Table 3: The top 7 leaderboard of the L2RPN WCCI 2020 Challenge among 50 participants.

Perhaps surprisingly, the performance of SMAAC\AS is no better than using primitive actions, although the hierarchical decision encourages to deviate from the initial topology. Without the afterstate representation, the critic was not able to learn a good action-value function due to the massive state and action spaces. The performance on the test scenarios is provided in Table 2. We provide qualitative analysis of how each agent behaves differently and how SMAAC remedies the hazardous power grid with a detailed example in Appendix A.4.

On the contrary, our method learns significantly fast and outperforms all the baselines, effectively combining the benefits of the hierarchical decision model and the afterstate representation. Finally, Table 3 shows the leaderboard in the L2RPN WCCI 2020 challenge, comparing the scores of top 7 participants.

5.3 Low-level rule design

In this section, we examine how the low-level policy affects the overall performance. (1) **RAND** gives priority to substations randomly that is predefined and unchanged during training. We implement this low-level agent to find out our high-level agent that is able to manage the power network on the poor low-level agent. (2) **CAPA** gives high priority to substations with lines under high utilization of their capacity, which applies action to substations that require urgent care. (3) **DESC** imposes a priority on large substations, i.e. many collected elements. A change to a large substation can be seen as making a large change in the overall topology with a single action, taking less time steps to get to the goal topology. (4) **OPTI** optimizes execution order by training, making the actor additionally output N_{sub} values that represent the priority of substations. As shown in Figure 4, except for RAND, all rules achieve similar performance. Especially, CAPA converges fast compared to OPTI and DESC, hence we use this low-level agent in the Section 5.2. The result shows that the poorly designed rule could lead to instability and degrade the performance. Therefore, an appropriate rule design is required for the successful management of the power grid.

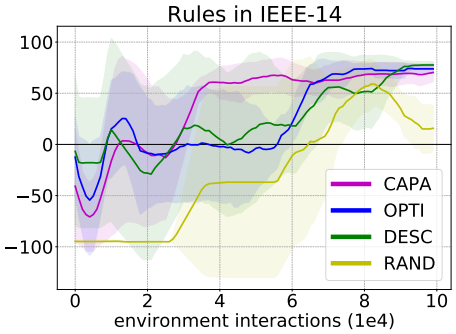

Figure 4: Comparison of 4 rules.

6 Conclusion

In this paper, we presented SMAAC, a deep RL approach demonstrated to be very effective for power grid management. SMAAC is an actor-critic algorithm that combines the afterstate representation with a hierarchical decision model. This is very important for power grid management modeled by Grid2Op, where actions are too primitive for effective exploration and many permutations of action sequences lead to identical changes in the power grid topology. In addition, naive explorations with primitive actions are subject to immediate failure due to the unique nature of power grid management. We empirically demonstrated that the presented method significantly outperforms several baselines in the real-world scale power grids, and ranked first in the latest international competition, L2RPN WCCI 2020 challenge. Our work shows the possibility of an intelligent agent that automatically operates the power grid several days without expert help.

REFERENCES

M. Alhazmi, P. Dehghanian, S. Wang, and B. Shinde. Power grid optimal topology control considering correlations of system uncertainties. In *2019 IEEE/IAS 55th Industrial and Commercial Power Systems Technical Conference (I CPS)*, pp. 1–7, 2019.

Payman Dehghanian, Yaping Wang, Gurunath Gurrala, Erick Moreno-Centeno, and Mladen Kezunovic. Flexible implementation of power system corrective topology control. *Electric Power Systems Research*, 128:79–89, 11 2015.

E. B. Fisher, R. P. O'Neill, and M. C. Ferris. Optimal transmission switching. *IEEE Transactions on Power Systems*, 23(3):1346–1355, 2008.

J. D. Fuller, R. Ramasra, and A. Cha. Fast heuristics for transmission-line switching. *IEEE Transactions on Power Systems*, 27(3):1377–1386, 2012.

A. Khodaei and M. Shahidehpour. Transmission switching in security-constrained unit commitment. *IEEE Transactions on Power Systems*, 25(4):1937–1945, 2010.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *International Conference on Learning Representations*, 2015.

Tu Lan, Jiajun Duan, Bei Zhang, Di Shi, Zhiwei Wang, Ruisheng Diao, and Xiaohu Zhang. Ai-based autonomous line flow control via topology adjustment for maximizing time-series atcs. *arXiv preprint arXiv:1911.04263*, 2019.

A. Marot, B. Donnot, S. Tazi, and P. Panciatici. Expert system for topological remedial action discovery in smart grids. *IET Conference Proceedings*, pp. 43 (6 pp.)–43 (6 pp.)(1), January 2018. URL `https://digital-library.theiet.org/content/conferences/10.1049/cp.2018.1875`.

Antoine Marot, Benjamin Donnot, Camilo Romero, Luca Veyrin-Forrer, Marvin Lerousseau, Balthazar Donon, and Isabelle Guyon. Learning to run a power network challenge for training topology controllers. *The Power Systems Computation Conference*, 2020.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Ofir Nachum, Shixiang (Shane) Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 31*, pp. 3303–3313. 2018.

Emilio Parisotto, H Francis Song, Jack W Rae, Razvan Pascanu, Caglar Gulcehre, Siddhant M Jayakumar, Max Jaderberg, Raphael Lopez Kaufman, Aidan Clark, Seb Noury, et al. Stabilizing transformers for reinforcement learning. In *Proceedings of The 37th International Conference on Machine Learning*, 2020.

Warren B. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality (Wiley Series in Probability and Statistics)*. Wiley-Interscience, USA, 2007.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *International Conference on Learning Representations*, 2016.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1–2):181–211, 1999.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pp. 5998–6008, 2017.

A. N. Venkat, I. A. Hiskens, J. B. Rawlings, and S. J. Wright. Distributed mpc strategies with application to power system automatic generation control. *IEEE Transactions on Control Systems Technology*, 16(6):1192–1206, 2008.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. Dueling network architectures for deep reinforcement learning. In *Proceedings of the 33th International Conference on Machine Learning*, pp. 1995–2003, 2016.

Z. Zhang, D. Zhang, and R. C. Qiu. Deep reinforcement learning for power system applications: An overview. *CSEE Journal of Power and Energy Systems*, 6(1):213–225, 2020.

C. Zhao, U. Topcu, N. Li, and S. Low. Design and stability of load-side primary frequency control in power systems. *IEEE Transactions on Automatic Control*, 59(5):1177–1189, 2014.

A Appendix

A.1 Environment Detail

Grid2Op provides a simulation for power grid operation in real-time over several days at a 5-minute time-resolution. There are 3 power grids, IEEE-5, IEEE-14, and L2RPN WCCI 2020 (a subgraph of IEEE-118), where each has different size.

State Space The state of the power grid consists of 12 features presented in Table 4. We use 5 features provided by the environment and 1 feature defined by us, which we consider enough to represent the current state of the grid; Active power, rho, topology configuration, time step overflow, maintenance, danger. The maintenance is a boolean vector representing whether a line is in maintenance, and the danger represent is also a boolean vector representing whether electricity flows of a line is larger than a predefined threshold δ_d. We use 0.9 for the threshold, and this threshold is same as the one we used for the hazardous state. Further details are provided in `grid2op.readthedocs.io/en/latest/observation.html`

Action Space The agent can apply actions on substations and lines to manage the power grid. The action on a substation, called *bus assignment*, assigns the elements in the substation to a busbar. The action on a line, called *line switch*, disconnects (both ends of line is assigned to neither bus) a line or reconnects a disconnected line. Let us define the number of lines in the power grid as N_{line}, the number of substations as N_{sub}, and the number of elements in ith substation as $Sub(i)$. Then at each time step the agent selects an action a_t from the action space $\mathcal{A}$ where $|\mathcal{A}| = N_{line} + 2^2 \times N_{line} + \sum_{i=0}^{N_{sub}} 2^{Sub(i)}$. [1]

Rule There are some rules that make the task more realistic and challenging in Grid2Op. Lines can be automatically disconnected due to overflow of current, i.e. if more current flows than a line can hold for 3 time steps, a line is automatically disconnected. There is cooldown time for each component, i.e. the agent cannot apply its action to the same component successively and it is reactivated after 3 time steps later. There is a stochastic event called maintenance that happens intermittently. During maintenance, a line is disconnected by force and it cannot be reconnected.

[1] Both ends of the line can be assigned to one of two buses in the substation, so $2^2 \times N_{line}$ is the number of reconnection. Each end of the line, generator, and load in the substation also can be assigned to one of two buses in the substation, so bus switching actions are $\sum_{i=0}^{N_{sub}} 2^{Sub(i)}$

Name	Type	Size	Description
Date	int	6	The current year, month, day, hour of day, minute of hour, day of week.
Active power	float	$N_{gen} + N_{load}$ $+2 \times N_{line}$	Active power magnitude.
Reactive power	float	$N_{gen} + N_{load}$ $+2 \times N_{line}$	Reactive power magnitude.
Voltage	float	$N_{gen} + N_{load}$ $+2 \times N_{line}$	Voltage magnitude.
Rho	float	N_{line}	The capacity of each power line, which is defined as ratio between current flow and thermal limit.
Topology Configuration	int	$N_{gen} + N_{load}$ $+2 \times N_{line}$	For each element (load, generator, ends of a line), it gives on which bus these elements is connected in its substation.
Line status	bool	N_{line}	The status of each line, whether it is connected or disconnected.
Time step overflow	int	N_{line}	The number of time steps each line is overflowed.
Time before cooldown line	int	N_{line}	How much cooldown time for each line is left. An agent cannot perform its action on lines with cooldown.
Time before cooldown sub	int	N_{sub}	How much cooldown time for each substation is left. An agent cannot perform its action on substations with cooldown.
Time next maintenance	int	N_{line}	The time of the planned maintenance. An agent cannot act on the lines with maintenance.
Duration next maintenance	int	N_{line}	The number of time steps that the maintenance will last.

Table 4: Details of features in the state

Score The performance of an agent can be evaluated by a score which consists of power loss penalty, failure penalty, and redispatching penalty. It is defined as :

$$Score = \sum_{t=0}^{t_{over}} (prod_t - load_t) + \sum_{t=t_{over}}^{t_{end}} penalty + \sum_{t=0}^{t_{over}} redispatch_t \tag{8}$$

where t_{over} is the time game over occurs, $prod_t$ is total amount of power supply by all generators, $load_t$ is total amount of power demand by loads. $prod_t - load_t$ stands for total amount of power loss. The failure penalty is given by the sum over the large constant $penalty$ for the remaining simulation time steps upon termination. The redispatching penalty is incurred when the agent do redispatching action, which controls generators to produce more or less electricity. Since redispatching action always incurs the additional penalty, we do not consider this action, so the redispatching penalty is always 0 in our case. Therefore, the goal of the agent is to operate the power grid both safely and efficiently by minimizing the failure penalty and the power loss penalty.

A.2 Model Architecture

Given the state $s = [\tau, x]$, we reshape x into $(M, \tilde{x})$, where $M \in \{0,1\}^{n \times n}$ is the adjacency matrix, and $\tilde{x} \in \mathbb{R}^{n \times k}$ is the node matrix with k features. The shared layers consisted of L_s GNN blocks that computes the node embedding of an input graph through transformer layers at their beginning. Given input nodes matrix $\tilde{x}$, a linear layer with ReLU activation increases input dimension k to embedding dimension k_s, which maps $\tilde{x} \in \mathbb{R}^{n \times d}$ to $H^0 \in \mathbb{R}^{n \times k_s}$. After the linear layer, L_s transformer layers follow. The input of the transformer block at the lth block is an embedding from the previous layer H^{l-1} and the adjacency matrix $\mathcal{A}$, $H^l = Transformer(H^{l-1}, A)$.

The actor's head consists of L_a transformer blocks and 2 linear layers. Given the final node embedding H^{L_e} from the shared layers, the transformer layers in actor's head takes it as the input and outputs node embedding $H^{L_a} \in \mathbb{R}^{n \times k_a}$. The first linear layer transforms 2D node embedding H^{L_a} to a vector node embedding $\mathbb{R}^n$ by reducing the embedding dimension k_a to 1, which is then concatenated with the current topology τ to form the state s and the next linear layers outputs mean and standard deviation of the normal distribution. We sample continuous values $g' \in \mathbb{R}^n$ from the normal distribution followed by tanh non-linearity and the desirable topology $g \in \{0, 1\}^n$ is constructed by assigning 1 to values in g' larger than predefined topology threshold δ_τ and 0 otherwise. We empirically find out that an agent without the threshold has difficulty learning in the large grid. However, an appropriate threshold helps stable learning and fast convergence.

The critic's head has a similar structure except for linear layers. There are L_c GNN blocks in the critic that also takes H^{L_e} and outputs $H^{L_c} \in \mathbb{R}^{n \times k_c}$. After transforming it into a vector node embedding $\mathbb{R}^n$ by a linear layer, g' is concatenated to H^{L_c}, and the following two linear layers take it as the input and outputs a scalar value. The overall architecture is shown in the Figure 2.

Although the agent observes all substations, we reduce the goal dimension n to $\tilde{n}$ by restricting controllable substation. As we mentioned in subsection 3.1, we do not consider disconnection or reconnection. Therefore, the agent only controls substations that have more than 2 elements since there are only two possible cases, elements on a same bus (connection) or elements on a different bus (disconnection). For L2RPN WCCI 2020 grid, the agent acts on substations that have more than 5 elements in order for fast convergence. As a result, the goal dimension is reduced from 21 to 16 in IEEE-5, from 57 to 42 in IEEE-14, and from 177 to 79 in L2RPN WCCI 2020.

A.3 Implementation Details

For all models, we use 6 state features, active power, rho, topology configuration, time step overflow, maintenance, and danger, where $\delta_d = 0.9$. Additionally the last 6 states of a history are stacked to represent the input state, since the difference between the state at time step t and $t+1$ is not significantly different in Grid2Op. Since the first decimal place of reward does not change significantly ($\frac{load_t}{prod_t}$ varies from 0.85 to 0.99 most of time), we transform it as $(\frac{load_t}{prod_t} \times 10 - 9) \times 0.1$ to use the second decimal place. Adam optimizer (Kingma & Ba, 2015) is used for training with $5e-5$ learning rate and 128 batch size. We perform grid search to find the best hyperparameters for each model.

SMAAC SMAAC is our proposed model, which learns the afterstate-value function on the goal space, namely topology configuration space. We use $L_s = 6$ GNN blocks with embedding dimension $k_s = 64/128$ for shared layers. For actor's head, we use $L_a = 3$ GNN blocks with embedding dimension $k_a = 64/128$. For critic's head, we use $L_c = 1$ GNN block with embedding dimension $k_c = 64/128$ followed by linear layers with $\frac{k_c + \tilde{n}}{4}$ hidden units. We use $\delta_\tau = 0/0.1/0.15$. In practice, the reward for the high-level policy in Equation 6 is not discounted, $r_{t:t+d} = \sum_{t'=t}^{t+d} r_{t'}$ and γ is used instead of γ^d following Nachum et al. (2018). For the competition, we use $k_s = 128$, $k_c = 128$, $k_a = 128$, $\delta_\tau = 0.35$, and τ is used as an extra input feature for the shared layers.

SMAAC\AS SMAAC\AS is a baseline, which learns the action-value function of desired relative change in hierarchical framework. The overall architecture is similar to SMAAC, but the critic takes both desired relative change and topology configuration to learn on state-action pairs. We use $\delta_\tau = 0/0.1/0.15$.

SAC SAC is a baseline, which learns the action-value function on the primitive bus assignment action space. We use $L_s = 6$ GNN blocks with embedding dimension $k_s = 64/128$ for shared layers. For actor's head, we use $L_a = 3$ GNN blocks with embedding dimension $k_a = 64/128$. And, we use softmax to output categorical distribution while utilize relaxed categorical distribution in training. For critic's head, we use $L_c = 1$ GNN block with embedding dimension $k_c = 32/64/128$ followed by concatenation with one-hot encoded action and one linear layer.

DDQN DDQN is a baseline, which learns the action-value function on the primitive bus assignment action space. It does not have separate actor but critic outputs $|\mathcal{A}|$ action-values. We

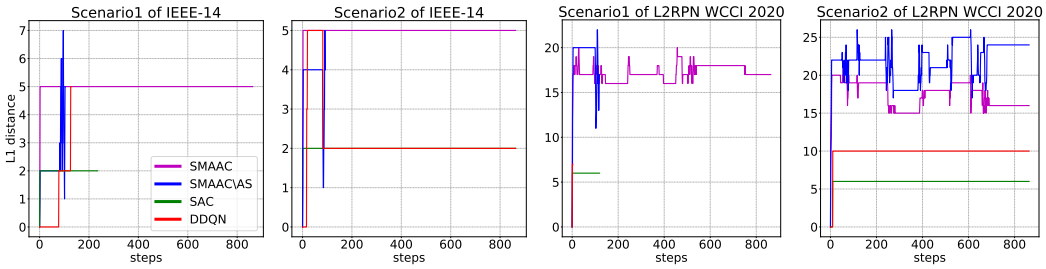

Figure 5: Deviation from the initial grid. We measure L1 distance between the initial topology and the current topology during rollout in 2 test scenarios for each grids.

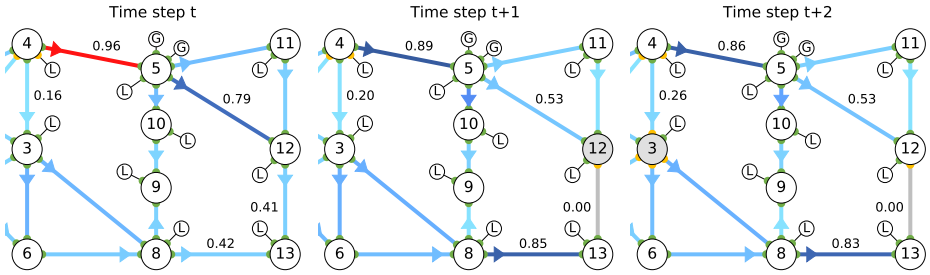

Figure 6: Illustration of a part of IEEE-14 managed by our agent. Large circle refers to a substation, small circle with G or L refers to a generator or load, and dots in green or yellow on a substation refers 2 busbars. The numbers beside lines are line usage ratio of current to line capacity and the darker color of the line means large ratio.

use $L_s = 6$ GNN blocks with embedding dimension $k_s = 64/128$ for embedding layers. Following it, the critic has $L_c = 1$ GNN blocks with embedding dimension $k_s = 64/128$. Then, it utilizes the technique used by dueling DQN, namely we compute action-values through both value network and advantage network.

A.4 Qualitative Analysis

In this section, we present qualitative analysis based on example behaviors of agents. Figure 5 indirectly shows how each agent behaves in 2 grids. In the medium grid, after our agent reaches to the certain grid, it keeps staying in that grid. We speculate that the agent finds the optimal grid during training where electricity could distribute evenly all the time so no further actions required. It is reasonable in Grid2Op where a single action can potentially destroy the grid. On the other hand, it shows different behavior in the large grid. It changes the topology configuration diversely to revert the hazardous situations since the agent can not find the grid such as the one in the medium grid.

SAC, which shows the best performance among the baselines, only changes a few of the topology configuration by executing only one or two actions in both grids, since the initial topology is strong local optima in Grid2Op. DQN also shows similar behavior, but it shows worse performance than the SAC, since it changes the initial grid more. Likewise, SAC+HIER that changes the topology the most shows the worst performance. It is extremely difficult to find the better topology than the initial one in Grid2Op without the effective exploration. The efficient learning together with the effective exploration is the key to successful management.

We further examine how SMAAC learned to revert a hazardous state back to the safe state. As shown in Figure 6, the line 4-5 (between substation 4 and 5) in t is in the hazardous situation. At time step $t+1$, our agent makes bus assignment change in the substation 12 by assigning line 12-13 to yellow busbar. As a result, some amount of electricity that flows in the line 4-5 moves to line 4-3 to meet the load demand in the substation 13 where electricity is supplied only from the line 8-13 due to the action in $t+1$. Then the last action at $t+2$ that changes the substation 3 further disperses electricity from the substation 4. In the end, a more balanced distribution is achieved.