Deep Reinforcement Learning Solutions for Energy Microgrids Management

Vincent François-Lavet V.FRANCOIS@ULG.AC.BE
David Taralla DTARALLA@ULG.AC.BE
Damien Ernst DERNST@ULG.AC.BE
Raphael Fonteneau RAPHAEL.FONTENEAU@ULG.AC.BE
Department of Electrical Engineering and Computer Science, University of Liege, Belgium

Abstract

This paper addresses the problem of efficiently operating the storage devices in an electricity microgrid featuring photovoltaic (PV) panels with both short- and long-term storage capacities. The problem of optimally activating the storage devices is formulated as a sequential decision making problem under uncertainty where, at every time-step, the uncertainty comes from the lack of knowledge about future electricity consumption and weather dependent PV production. This paper proposes to address this problem using deep reinforcement learning. To this purpose, a specific deep learning architecture has been designed in order to extract knowledge from past consumption and production time series as well as any available forecasts. The approach is empirically illustrated in the case of a residential customer located in Belgium.

1. Introduction

An electricity microgrid is an energy system consisting of local electricity generation, local loads (or energy consumption) and storage capacities. In this paper, we consider microgrids that are provided with different types of storage devices in order to be able to address both short- and long-term fluctuations of electricity production using photovoltaic (PV) panels (typically, batteries for short-term fluctuations, and hydrogen/fuel cells for long-term fluctuations). Distinguishing short- from long-term storage is mainly a question of cost: batteries are currently too expensive to be used for addressing seasonal variations. Energy microgrids face a dual stochastic-deterministic structure: one of the main challenge to meet when operating microgrids is to find storage strategies capable of handling uncertainties related to future electricity production and consumption; besides this, microgrids also have the characteristics that their dynamics deterministically reacts to storage management actions.

In this paper, we propose to design a storage management strategy which exploits this characteristic. We assume that we have access to: (i) an accurate simulator of the (deterministic) dynamics of a microgrid and (ii) time series describing past load and production profiles, which are realizations of some unknown stochastic processes. In this context, we propose to design a deep Reinforcement Learning (RL) agent (Mnih et al. (2015)) for approximating the optimal strategy through interaction with the environment. The deep RL algorithm proposed in this paper has been specifically designed to the setting which is original in the sense that the environment is partly described with a deterministic simulator (from which we can generate as much data as necessary), and partly with a limited batch of real stochastic time series (load and production). Unlike Kuznetsova et al. (2013), our specific deep Neural Network (NN) architecture is built upon a large continuous non-handcrafted feature space that uses convolutional layers to extract meaningful features from time series (see e.g. Szegedy et al. (2015)). Compared to the approach in Mnih et al. (2015), we propose a validation strategy which periodically evaluates how well the policy performs on unseen time series to ensure that the agent does not overfit on the limited training data. Finally, our approach also aims at minimizing sources of errors that may appear in other related approaches: for instance, positive bias generated when learning from imitation of optimal solutions (Aittahar et al. (2015)) or e.g. errors associated with scenario aggregation in stochastic programming (Mohammadi et al. (2014)).

This paper is organized as follows: Section 2 describes the deep RL framework. Section 3 details the microgrid benchmark. Section 4 introduces our Deep RL structure dedicated to microgrid management, as well as empirical results corresponding to the case of a residential customer located in Belgium. Section 5 concludes.

2. Deep reinforcement learning solutions for sequential decision making

Optimally operating a microgrid can be formalized as a partially observable Markov decision process, where the microgrid is considered as an agent that interacts with its environment. In order to approach the Markov property, the state of the system $s_t \in \mathcal{S}$ is made up of an history of features of observations $O_t^i, i \in \{1, \ldots, N_f\}$, where $N_f \in \mathbb{N}$ is the total number of features. Each O_t^i is represented by a sequence of punctual observations over a chosen history of length h^i: $O_t^i = [o_{t-h^i+1}^i, \ldots, o_t^i]$ (the history length may depend on the feature). At each time step, the agent observes a state variable s_t, takes an action $a_t \in \mathcal{A}$ and moves into a state $s_{t+1} \sim P(\cdot | s_t, a_t)$. A reward signal $r_t = \rho(s_t, a_t, s_{t+1})$ is associated to the transition (s_t, a_t, s_{t+1}), where $\rho : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function. We then define the γ-discounted optimal Q-value function:

$$Q^*(s,a) = \max_{\pi} \underset{s_{t+1},\, s_{t+2},\, \ldots}{\mathbb{E}} \left[\sum_{k=t}^{\infty} \gamma^{k-T} r_k | s_t = s, a_t = a, \pi \right].$$

We propose to approximate Q^* using a Neural Network (NN). We denote by $Q(\cdot, \cdot; \theta_k)$ the so-called Q-network. NNs offer generalization properties that are adapted to high-dimensional sensory inputs such as temporal series. The NN parameters θ_k may be updated using stochastic gradient descent (or other related techniques) by sampling batches of transitions (s, a, r, s') in a replay memory, updating the current value $Q(s, a; \theta_k)$ towards a target value $Y_k^Q = r + \gamma \arg\max_{a' \in A} Q(s', a'; \theta_k^-)$ where θ_k^- refers to parameters from some previous Q-network called the target Q-network as introduced in Mnih et al. (2015). When using the squared-loss, a Q-learning update is obtained as follows:

$$\theta_{k+1} = \theta_k + \alpha (Y_k^Q - Q(s, a; \theta_k)) \nabla_{\theta_k} Q(s, a; \theta_k) \tag{1}$$

where α is a scalar step size called the *learning rate*.

3. Electricity microgrid: benchmark description

First, note that the microgrid model described hereafter is fully described in François-Lavet et al. (2016). We denote the storage operation state of the microgrid by $s_t^{MG} \in \mathcal{S}^{MG}$: it describes the amount of energy in the storage devices. The amount of energy in the battery is denoted by s_t^B[Wh] $\in S_t^B$ and the amount of energy in the hydrogen tank is denoted by $s_t^{H_2}$[Wh] $\in \mathcal{S}_t^{H_2}$. We introduce x^B $[Wh]$ (resp. x^{H_2} $[W_p]$) as the battery (resp. hydrogen) storage sizing. The variable η^B (resp. ζ^B) denotes the battery discharge (resp. charge) efficiency. Similarly, the electrolysis and fuel cells efficiencies are respectively denoted by η^{H_2} (when storing energy) and ζ^{H_2} (when delivering energy). At every time step, an action $a_t = [a_t^{H_2}, a_t^B] \in \mathcal{A}_t$ is applied on the system, where $a_t^{H_2}$ is the amount of energy transferred into (if positive) or out of (if negative) the hydrogen storage device (H_2), similarly a_t^B is the amount of energy transferred into or out of the battery (B). We have, $\forall t \in \mathcal{T}$: $\mathcal{A}_t = \left([-\zeta^B s_t^B, \frac{x^B - s_t^B}{\eta^B}]\right) \times \left([-\zeta^{H_2} s_t^{H_2}, \infty[\cap [-x^{H_2}, x^{H_2}]\right)$ which expresses the fact that the bounds on the power flows of the storing devices are, at each time step $t \in \mathcal{T}$, the most constraining among the ones induced by the charge levels and the power limits. The battery dynamics is given by: $s_{t+1}^B = s_t^B + \eta_t^B a_t^B$ if $a_t^B \geq 0$ and $s_{t+1}^B = s_t^B + \frac{a_t^B}{\zeta_t^B}$ otherwise. Similarly, the hydrogen dynamics is given by: $s_{t+1}^{H_2} = s_t^{H_2} + \eta_t^{H_2} a_t^{H_2}$ if $a_t^{H_2} \geq 0$ and $s_{t+1}^{H_2} = s_t^{H_2} + \frac{a_t^{H_2}}{\zeta_t^{H_2}}$ otherwise.

The reward function of the system corresponds to the instantaneous operational revenues r_t at time $t \in \mathcal{T}$. We now introduce three quantities that are prerequisites to the definition of the reward function: (i) ϕ_t $[Wh] \in \mathbb{R}^+$ is the electricity generated locally by the PV installation, (ii) d_t $[Wh] \in \mathbb{R}$ denotes the net electricity demand, which is the difference between the local consumption c_t and the local production of electricity ϕ_t, (iii) δ_t $[Wh] \in \mathbb{R}$ represents the power balance within the microgrid, taking into account the contributions of the net electricity demand and the charge or discharge of the storage devices: $\delta_t = -a_t^B - a_t^{H_2} - d_t$. The instantaneous reward signal r_t is obtained by adding the revenues generated by the hydrogen production r^{H_2} with the penalties r^- due to the value of loss load: $r_t = r(a_t, d_t) = r^{H_2}(a_t, d_t) + r^-(a_t, d_t)$. The penalty r^- is proportional to the total amount of energy that was not supplied to meet the demand: $r^-(a_t, d_t) = k\delta_t$ when $\delta_t < 0$ and null otherwise (k is the cost endured per Wh not supplied within the microgrid), while r^{H_2} is given by: $r^{H_2}(a_t, d_t) = k^{H_2} a_t^{H2}$ (k^{H_2} is the revenue/cost per Wh of hydrogen produced/used)

From the series of rewards (r_t), we obtain the operational revenues over year y defined as follows: $M_y = \sum_{t \in \tau_y} r_t$ where τ_y is the set of time steps belonging to year y. Optimizing the operation of the microgrid requires to determine a sequential decision making strategy that leads to the maximization of M_y. Note that in the microgrids literature, researchers often use the overall Levelized Energy Cost (LEC) criterion, which is an economic assessment of the cost that covers all the expenses over the lifetime of the microgrid (i.e. initial investment, operation, maintenance and cost of capital): $LEC_r = \frac{I_0 + \sum_{y=1}^{n} \frac{M_y}{(1+r)^y}}{\sum_{y=1}^{n} \frac{\epsilon_y}{(1+r)^y}}$ where n denotes the lifetime of the system in years; I_0 corresponds to the initial investment; M_y represents the operational expenses in the year y; ϵ_y is the electricity consumption in the year y; r denotes the discount rate (either interest rate or cash flow discount).

4. Applying deep reinforcement learning for managing microgrids

We consider the case of a residential electricity consumer (average of 18kWh/day) located in Belgium operating an off-grid microgrid. The cost k endured per kWh not supplied within the microgrid is set to 2 €/kWh. Other microgrid parameters are taken from François-Lavet et al. (2016). Three different cases (resulting in different state vectors) are considered:

(i) a base case with minimal information available to the agent:

$$s_t = \big[[c_{t-h^c}, \ldots, c_{t-1}], [\phi_{t-h^p}, \ldots, \phi_{t-1}], s_t^{MG}\big]$$

where $h^c = 12h$ and $h^p = 12h$ are the lengths of the time series considered as input to the neural network (consumption and production respectively);
(ii) the case where information on the season is provided:

$$s_t = \big[[c_{t-h^c}, \ldots, c_{t-1}], [\phi_{t-h^p}, \ldots, \phi_{t-1}], s_t^{MG}, \zeta_s\big]$$

where ζ_s is the smallest number of days to the solar solstice (21^{st} of June) which is then normalized into [0,1];
(iii) the case where accurate production forecasting is available:

$$s_t = \big[[c_{t-h^c}, \ldots, c_{t-1}], [\phi_{t-h^p}, \ldots, \phi_{t-1}], s_t^{MG}, \zeta_s, \rho_{24}, \rho_{48}\big]$$

where ρ_{24} (resp. ρ_{48}) is the (known) solar production for the next 24 hours (resp. 48 hours).

4.1 Neural network architecture

We propose a Neural Network (NN) architecture where the inputs are provided by the state vector, and where each separate output represents the Q-values for each discretized action. Possible actions a are whether to charge or discharge the hydrogen storage device with the assumption that the batteries handle at best the current demand (avoid any value of loss load whenever possible). We consider three discretized actions : (i) discharge at full rate the hydrogen storage, (ii) keep it idle or (iii) charge it at full rate.

The NN processes time series thanks to a set of convolutions with 16 filters of 2×1 with stride 1 followed by a convolution with 16 filters of 2×2 with stride 1. The output of the convolutions as well as the other inputs are then followed by two fully connected layers with 50 and 20 neurons and the output layer. The activation function used is the Rectified Linear Unit (ReLU) except for the output layer where no activation function is used.

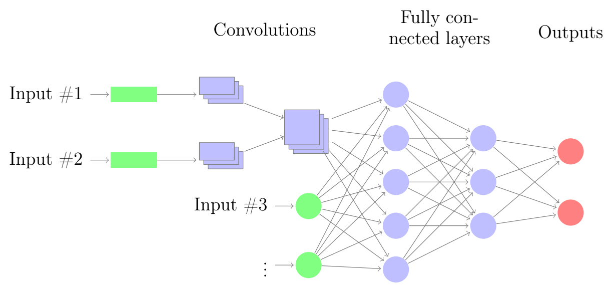


Figure 1: Sketch of the structure of the NN architecture. The NN processes time series thanks to a set of convolutional layers. The output of the convolutions as well as the other inputs are followed by fully connected layers and the ouput layer. Architechtures based on LSTMs instead of convolutions obtain close results and the reader is welcome to experiment with the source code.

4.2 Splitting times series to avoid overfitting

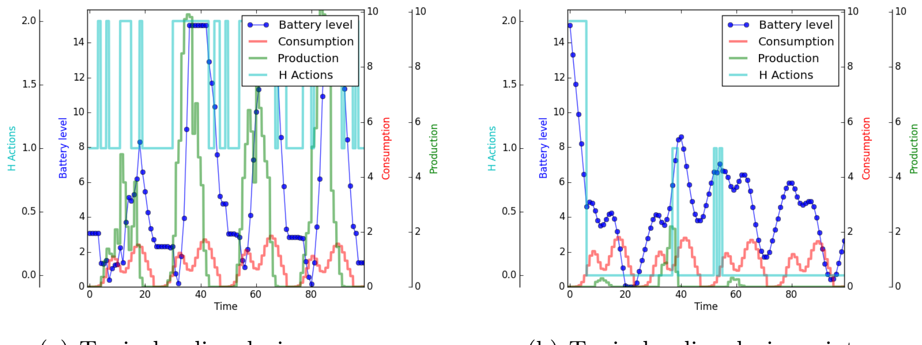
We consider the case where the agent is provided with two years of actual past realizations of (c_t) and (ϕ_t). In order to avoid overfitting, these past realizations are split into a training environment ($y = 1$) and a validation environment ($y = 2$). The training environment is used to train the policy while the validation environment is used at each epoch to estimate how well the policy performs on the undiscounted objective M_y and selects the final NN. The final NN is then used in a test environment ($y = 3$) to provide an independent estimation on how well the policy performs.

4.3 Training

By starting with a random Q-network, we perform at each time step the update given in Eq. 1 and, in the meantime, we fill up a replay memory with all observations, actions and rewards using an agent that follows an ϵ-greedy policy s.t. the policy $\pi(s) = \max_{a \in \mathcal{A}} Q(s, a; \theta_k)$ is selected with a probability $1 - \epsilon$, and a random action (with uniform probability over actions) is selected with probability ϵ. We use a decreasing value of ϵ over time. During the validation and test phases, the policy $\pi(s) = \max_{a \in \mathcal{A}} Q(s, a; \theta_k)$ is applied (with $\epsilon = 0$). As discussed in François-Lavet et al. (2015), we use an increasing discount factor along with a decreasing learning rate through the learning epochs so as to enhance learning performance.

4.4 Results and discussions

We consider a robust microgrid sizing provided by François-Lavet et al. (2016). The size of the battery is $x^B = 15kWh$, the instantaneous power of the hydrogen storage is $x^{H_2} = 1.1kW$ and the peak power generation of the PV installation is $x^{PV} = 12kW_p$. We first run the base case with minimal information available. The selected policy is based on the best validation score. The typical behaviour of the policy is illustrated in Figure 2 (test data). Since the microgrid has no information about the future, it builds up (during the night) a sufficient reserve in the short-term storage device so as to be able to face the next day consumption without suffering too much loss load. It also avoids wasting energy (when the short term storage is full) by storing in the long-term storage device whenever possible.

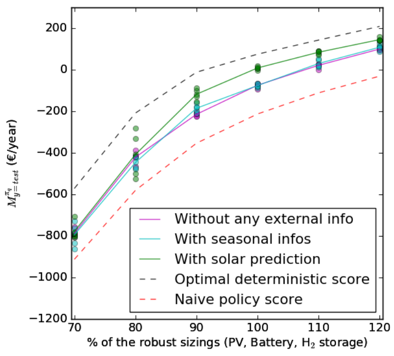


(a) Typical policy during summer (b) Typical policy during winter

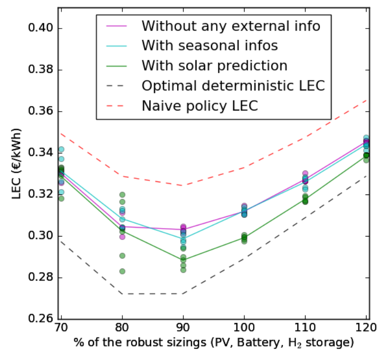
Figure 2: Computed policy with minimal information available to the agent. $H\ action = 0$ means discharging the hydrogen reserve at maximum rate; $H\ action = 1$ means doing nothing with the hydrogen reserve; $H\ action = 2$ means building up the hydrogen reserve at maximum rate.

We now investigate the effect of providing additional information to the agent. We report in Figure 3(a) the operational revenue on the test data $M_y^{\pi_q}$ for the three cases as a function of a unique percentage of the initial sizings x^B, x^{H_2}, x^{PV}. For each configuration, we run the process five times with different seeds. We first observe that the dispersion in the revenues is higher for small microgrids: the operation being more challenging in such cases, small differences in the decision process have a larger impact. Second, it can be observed that any useful information added as input to the agent helps improving the policy, such as accurate information about the production profile. Similarly, additional data on the consumption profile would help to further improve the policy π_q. This data could for instance take the form of the current week day (1 to 7) in order to model the case where a residential customer would consume, on average, more energy during some particular days of the week.

We can also plot the LEC obtained as a function of a unique percentage of the initial sizings x^B, x^{H_2}, x^{PV}. The LEC is calculated with the assumption that the operational revenue obtained for the test data is the same over the lifetime of the microgrid.



(a) Operational revenue



(b) LEC

Figure 3: Operational revenue and LEC (test data) function of the sizings of the microgrid. The optimal deterministic operation is the one obtained by solving the problem with the assumption of perfect knowledge of the whole future with the method described in François-Lavet et al. (2016). The Naive policy operation is obtained by optimizing the thresholds at which to discharge and charge the hydrogen storage based on the level of energy in the battery (through grid search on rollouts in the validation environment).

5. Conclusion

This paper has introduced a deep reinforcement learning architecture for addressing the problem of operating an electricity microgrid in a stochastic environment. The proposed approach is original in the overall validation process. Experimental results illustrate the fact that the NN representation of the value function efficiently generalizes the policy to situations corresponding to unseen configurations of electricity demand and solar irradiance. Future works include the extension of the microgrid simulator, in particular by increasing the diversity of electricity production and storage technologies. It would also be of interest to investigate the case where several microgrids interact with each other and with the main utility grid.

Data and source code

PV production and consumption profiles as well as main microgrid parameters can be found at `http://deer.readthedocs.io/en/master/user/environments/two_storages.html`. Source code is available at `https://github.com/VinF/deer`.

Acknowledgement

The authors thank the Walloon Region which has funded this research in the context of the BATWAL project. We also acknowledge the use of the Python library Theano (Theano Development Team (2016)).

The authors thanks the "Consortium des équipements de Calcul Intensif (CECI)", funded by the Fonds de la Recherche Scientifique de Belgique (F.R.S.-FNRS) under Grant No. 2.5020.11 for providing the computational resources needed for carrying out this research.

References

S Aittahar, V François-Lavet, S Lodeweyckx, D Ernst, and R Fonteneau. Imitative learning for online planning in microgrids. In *Data Analytics for Renewable Energy Integration*, pages 1–15. Springer, 2015.

V François-Lavet, R Fonteneau, and D Ernst. How to discount deep reinforcement learning: Towards new dynamic strategies. *arXiv preprint arXiv:1512.02011*, 2015.

V François-Lavet, Q Gemine, D Ernst, and R Fonteneau. Towards the minimization of the levelized energy costs of microgrids using both long-term and short-term storage devices. *Smart Grid: Networking, Data Management, and Business Models*, pages 295–319, 2016.

E Kuznetsova, Y Li, C Ruiz, E Zio, G Ault, and K Bell. Reinforcement learning for microgrid energy management. *Energy*, 59:133–146, 2013.

V Mnih, K Kavukcuoglu, D Silver, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

S Mohammadi, S Soleymani, and B Mozafari. Scenario-based stochastic operation management of microgrid including wind, photovoltaic, micro-turbine, fuel cell and energy storage devices. *International Journal of Electrical Power & Energy Systems*, 54:525–535, 2014.

C Szegedy, W Liu, Y Jia, P Sermanet, S Reed, D Anguelov, D Erhan, V Vanhoucke, and A Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016. URL `http://arxiv.org/abs/1605.02688`.