Energy Management for Microgrids: a Reinforcement Learning Approach

Tanguy Levent
LPICM
École polytechnique, CNRS, IP. Paris
tanguy.levent@polytechnique.edu

Philippe Preux
CRIStAL
Université de Lille, CNRS
philippe.preux@inria.fr

Erwan Le Pennec
CMAP
École polytechnique, CNRS, IP. Paris
erwan.le-pennec@polytechnique.edu

Jordi Badosa
LMD-IPSL
École polytechnique, ENS PSL , Sorbonne, CNRS
jordi.badosa@lmd.polytechnique.fr

Gonzague Henri
Total S.A.
gonzague.henri@total.com

Yvan Bonnassieux
LPICM
École polytechnique, CNRS, IP. Paris
yvan.bonnassieux@polytechnique.edu

***Abstract*—This paper presents a framework based on reinforcement learning for energy management and economic dispatch of an islanded microgrid without any forecasting module. The architecture of the algorithm is divided in two parts: a learning phase trained by a reinforcement learning (RL) algorithm on a small dataset and the testing phase based on a decision tree induced from the trained RL. An advantage of this approach is to create an autonomous agent, able to react in real-time, considering only the past. This framework was tested on real data acquired at Ecole Polytechnique in France over a long period of time, with a large diversity in the type of days considered. It showed near optimal, efficient and stable results in each situation.**

***Index Terms*—Microgrid, Energy Management System, Agent Based, Decision Tree, Reinforcement Learning, Q-Learning**

I. Introduction

In order to adapt to climate change, the electricity sector will have to evolve toward a smart and decarbonized grid [1]. In this context, we saw large integration of Distributed Energy Resources (DERs) over the last past years including renewable energy sources and storage units [2]. Researchers have found that a large penetration of renewable energy could weaken the grid, eventually causing blackouts, due to their intermittent nature. To alleviate this problem, microgrids have been pushed as one of the possible solutions [3]. A microgrid is a single small scale power entity that can be connected to the main grid, from a single home to a small town, and can provide several benefits to utilities: maintaining the balance of the utility grid, reducing the peaks, reducing periods of load variability, enhancing the power quality and reliability services and decrease the feeder losses. Nevertheless, many challenges must be overcome to deploy microgrids at scale. Among those challenges [4], this paper focuses on managing the power dispatch in order to minimize the total cost operations, while maintaining the grid stability. As this study focuses on the energy management optimization, the tertiary control level of the hierarchical architecture, as proposed in [5]–[7] is considered.

In the past few years, machine learning (ML), a branch of artificial intelligence, became very popular and showed promises in a large number of areas. Reinforcement Learning is a branch of ML which aims at developing agents that learn to solve a task by interacting with their environment [8]. RL has already been proposed for microgrid in energy trading market [9], for distributed generation (DG) agents learning [10], for energy management [11]–[16]. This paper proposes a reinforcement learning algorithm combined with a supervised learning algorithm to improve the decision making for microgrid power economic dispatch. This specific approach has not been studied before. It does not require a large set of data; no forecast is needed; it is also a model-free algorithm that exhibits short computation time.

The paper is organized as follows: in Sec. II, the microgrid case study is presented and the objective function, constraints and hypothesis are defined. Sec. III introduces the RL approach and in Sec. IV, we describe how a decision tree is induced from what has been learned by RL. Sec. V presents the experimental results. Finally, Sec. VI concludes and draws on future lines of work.

II. Design and Modeling of the Microgrid

A simple microgrid architecture is considered with a small number of DER. Despite its simplicity, it is well suited to evaluate the performance of the proposed approach. The data used is coming from the following sources: consumption measurements from a tertiary building (Drahi-X Novation Center) and measurements from a photovoltaic test bench from SIRTA atmospheric observatory [17], both based on the Campus of Ecole polytechnique, in Palaiseau, France. This paper is our first step in a top-down logical approach for future works.

A. Microgrid Description

The model is an islanded microgrid consisting of a solar array (PV), a set of batteries, a diesel generator (genset) and building loads. The different units used for this case study are defined as:

- PV: a 15 kWp installation with mono-Si photovoltaic panels. The data was scaled from the measurements of one PV panel installed at SIRTA.
- Genset: a diesel generator of 20 kW.
- Batteries: Two batteries of 30 kWh of usable storage capacity.
- Loads: Electricity demand related to an office-like use, with the main load due to heating/cooling, electric plugs and lights.

B. Problem Statement and Objective function

The goal of this study is to minimize the operational costs of the entire microgrid system while meeting the demand at all times. The generators described in the previous section are managed by a supervisor, referred to as the Energy Management System (EMS). Having a smart and optimized EMS is necessary to reduce costs. The algorithm proposed here does not use a forecaster to generate a day ahead planning of the microgrid generators. Instead, each decision is taken instantly regarding the information given by the microgrid. For this reason, this paper does not fall into the Unit Commitment problem but respects the same rules like keeping the power balance between the supply and the demand. The EMS minimizes the following objective function:

$$J_{obj} = \sum_{t=0}^{T} |P_B(t)|\, m + P_G(t)q + P_C(t)c \tag{1}$$

P_B represents the charge or discharge power of the battery. The batteries are related to an approximate fixed cost denoted m [18]. P_G is the power output of the diesel generator, and its cost is considered as a linear function, represented by the variable q. Finally P_C and c are the power and the cost of a crash due to bad management. The time step is defined by t and the time period by T.

C. Constraints and Hypothesis

The main constraint is relative to the power balance in the microgrid which must be satisfied at all time:

$$P_{PV}(t) + P_B(t) + P_G(t) = P_L(t) \tag{2}$$

where $P_{PV}(t)$ is the power produced by the solar panels and $P_L(t)$ is the power demand, at time step t. A simplified dynamical battery storage is modeled as:

$$E_{Bcap}(t) = E_{Bcap}(t-1) - P_B(t)\Delta t \tag{3}$$

with E_{Bcap} refers to the battery energy capacity and $P_B\Delta t$ to the charge or discharge power (can be a positive or negative value). Furthermore, the battery capacity must stay within its limits at any time:

$$E_{Bmin} \leq E_{Bcap}(t) \leq E_{Bmax} \tag{4}$$

In addition, the genset must deliver an output power lower or equal than its maximum capabilities:

$$0 \leq P_G(t) \leq P_{Gmax} \tag{5}$$

knowing that if $P_G(t)$ is equal to zero, the genset is considered turned off. Finally the study manages a net demand P_{Net}, which is the difference between the power output of the solar panels and the consumption. The excess or deficit following at each step is the amount of power to supervise:

$$P_{Net}(t) = P_{PV}(t) - P_L(t) \tag{6}$$

III. Reinforcement Learning to control an EMS

Reinforcement learning [8] is based on the idea that an agent needs to interact with its environment to learn the optimal control policy. To be more specific, a learning agent (algorithm) that behaves sequentially along time t is considered. At each t, the agent perceives the state of its environment $s_t \in \mathcal{S}$, decides on an action $a_t \in \mathcal{A}$ and executes. Subsequently, it observes a return r_t and the new state of the environment s_{t+1}. The return is defined by a function that maps a state/action pair to the expected return when action a is emitted in state s: $\mathcal{R}(s,a) = \mathbb{E}[r(s,a)]$. The agent has to learn how to act in order to fulfill a task, that is to optimize a certain objective function. The most commonly used function is $\sum_{t\geq 0}\gamma^t r_t$, with $\gamma \in]0,1[$. Maximizing this function implies maximizing the return in the future. γ controls the extent to which we consider the future: the larger γ, the longer term consequences of the current actions are considered. If γ is small, γ^t quickly vanishes, which leads to disregard mid and long term consequences of current actions. The environment is supposed stochastic and Markovian: the next state s_{t+1} depends only on the current state and the current action $\mathcal{P}(s_{t+1}, s_t, a_t) = \mathbb{P}[s_{t+1}|s_t, a_t]$. The state of the environment perceived by the agent has to contain the necessary information to determine the best action to perform in the current state. The mapping state/best action is learned by repeated interaction between the agent and its environment in a trial-and-error fashion. In this paper, our goal is to design such a learning agent able to learn to control a microgrid. A microgrid is a stochastic, dynamic, sequential, continuous and partially observable environment. Dynamic means that the environment may change between two decision steps. Continuous because the set of possible actions and states are continuous. Partial observability means that the assumption that the agent has access to the true state of the environment can not be met here. Then, the agent has to rely on some information which is typically not enough to decide deterministically on the best action to perform. This is a typical situation when tackling a real problem with RL: we know the agent does not observe the full information, but we work as if it did. In summary, an RL agent solves a Markov Decision Problem (MDP) defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ in which $\mathcal{P}$ and $\mathcal{R}$ are unknown. Now, we are showing how we model the EMS with RL: the learning agent will learn how to control a microgrid. For this purpose, we define the different elements of this tuple.

A. States

The agent uses states to perceive its environment. The state design is decisive in the performance. The state space should be as small as possible in order to be easily manageable. It should contain all the information to choose the best action to perform. In our study, the state is defined as: $s = (P_{Net}, P_{Bcap})$. Only the net demand and the batteries capacity are considered to describe the microgrid. In this study, the state space is discretized at hourly steps.

B. Actions

The set of actions $\mathcal{A}$ considered in this study is:

- action 1 = Charge: batteries charge
- action 2 = Discharge: batteries discharge
- action 3 = Genset: genset produces electricity
- action 4 = Idle

C. Reward function

The return characterizes the current achievement of the task and also constraints. It is represented as a real value. In this study, the current reward is associated with the cost of the generator used to meet the net demand P_{Net}. Each generator has a distinct cost. A cost is also affected to the violation of the constraints. This value does not represent the true cost of an outage. It is used as a tool to penalize the agent for causing an outage. The Eq. (7) represents the reward conditional function:

$$r(s,a) = \begin{cases} -m * P_{Net}, & \text{if charge or discharge the battery} \\ -q * P_{Net}, & \text{if power produced by the genset} \\ -c * P_{Net}, & \text{if the constraints are not respected} \\ 0, & \text{if do nothing} \end{cases} \tag{7}$$

D. Q-Learning: a reinforcement learning algorithm

Q-learning is an algorithm to solve a RL problem, equivalent to computing its optimal policy. A policy assigns a probability distribution to each state on the set of actions. In the case of the MDP we consider in this paper, it is well-known that the optimal policy is a deterministic mapping from the set of states to the set of actions: in each state, there is one best action (or several strictly equivalent optimal actions). This policy is denoted π, $\pi(s)$ being the best action for state s. A key concept is the "value" of a state-action pair which formalizes the agent's benefit to select a certain action a in a state s, with regards to the optimization of the objective function, that is the fulfillment of its task [19]. This value is denoted $Q(s,a)$. This value depends on the policy π followed by the agent, hence $Q(s,a)$ is really $Q(s,a,\pi)$ usually denoted $Q^\pi(s,a)$ to emphasize the different nature of the parameters of Q. More formally, we have: $Q^\pi(s,a) = \mathbb{E}[r(s,a) + \gamma \max_{a'} Q^\pi(s',a')]$ known as the Bellman equation and s' represents the next state. Let us denote the optimal policy π^* and its Q function Q^*. We have $\pi^*(s) = \arg\max_a Q^*(s,a)$. Q^* can be learned iteratively through a stochastic approximation similar to a gradient descent algorithm which main iteration is:

$$\begin{aligned} Q(s_t,a_t) =& Q(s_t,a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1},a) \\ & - Q(s_t,a_t)] \end{aligned} \tag{8}$$

The series Q converges to Q^* under suitable assumptions. Eq. (8) is the essence of the Q-Learning algorithm. The learning rate α represents how much we are adjusting the new value of $Q(s,a)$. In this work, the estimate of Q is stored in a table. This Q-table is initialized randomly. Then, as the agent interacts with its environment, it updates the Q values with Eq. (8). Gradually, the agent will learn and will converge to the Q value of the optimal policy. For the task at hand, we will repeat several such episodes going from the initial state to a terminal state. Q-Learning is expressed more precisely in algorithm 1. The last important point concerns the choice of the action. As said above, RL is based on trial-and-error. Hence, initially, the agent has to try the various actions (explore the set of actions) and, as it learns, focus more and more on seemingly best actions (exploit the acquired knowledge). There are different methods to deal with action selection. The chosen method is called the "ϵ-greedy method". After enough iterations, the learning agent converges toward a good policy.

Algorithm 1 Q-Learning.

```
Set hyperparameters : α ∈ [0, 1], ε > 0
Initialize Q(s, a), ∀(s, a) ∈ S × A.
for each episode do
    Initialize the agent (s_0); t ← 0
    while Terminal state not reached do
        Choose a_t for s_t using Q
        Emit action a_t, observe r_t, s_{t+1}
        Update Q using (8)
    end while
end for
```

E. Q-transfer for optimizing the speed convergence

Q-Learning is used to provide the policy to manage the microgrid: it is trained on some past days management policies. The classical approach is to aggregate the past training days over one large episode to iteratively find Q^*. In this case, a Q-table is initialized and a Q_{Final} output table is created as a one direct process. In this paper, this large episode is split into several daily episodes. A Q-table is also initialized but instead of training over one set of days at the same time, the agent learns on a first day, provides a $Q_{Intermediate}$ table to a new day of training, and so on and so forth until to obtain a Q_{Final} table. At each training day, the hyperparameters α and ϵ are initialized. The term "Q-transfer" refers to the fact of passing a Q-table throughout daily episodes. We observe experimentally that the convergence is speed up 10 times using Q-transfer.

IV. Decision Tree to generalize experiences

At the end of the training phase, the Q_{Final} table is processed to extract the policy that will be used to control the EMS microgrid, $\pi_{Final}(s) = \arg\max_a Q_{Final}(s,a)$. This

π_{Final} is used to train a decision tree. The goal is to generalize from the experience. Some states and state-action pairs have not been seen during the training. The generalization is important to improve the performance of the policy controlling the microgrid EMS. A decision tree is a non-parametric supervised learning algorithm for regression or classification tasks. The training data consists of input/feature (state) and output/target (action) variables used to create rules in order to design a prediction model. The model is created by partitioning the training dataset and a prediction model is made at each split. The decision tree looks like a flowchart diagram where the terminal nodes represent the target decisions, and the internal nodes are conditions on the state environment values. We use a CART decision tree algorithm for this study [20]. CART builds a binary tree construction, which means that at each node only two branches are created. Fig. 1 illustrates a CART decision tree example obtained after training on Q_{Final}. Each "False" condition results in a terminal node, with the action to take, except for the last node condition where both results in a decision making.

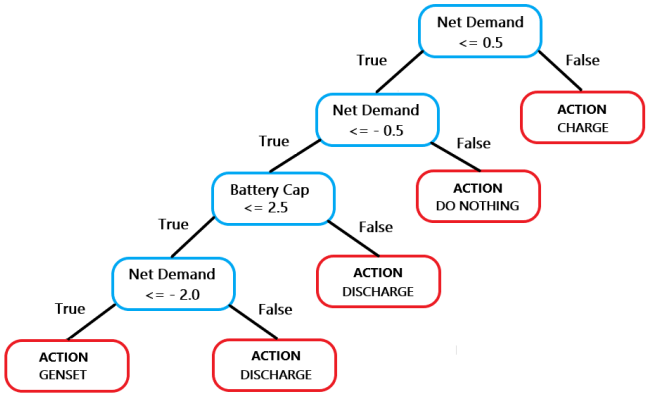

Fig. 1. A decision tree extracted from the Q-Table.

V. Experiments

We have collected data during 52 weeks, starting on 7/15/2016. The following costs are considered, $m = 0.5$, $q = 1.5$ and $c = 10$. The next two subsections will discuss the performance of the control algorithm. First, the training phase is validated, then testing results are shown. The training phase consists in learning from past experience, finally providing a decision tree. The testing phase consists in using this decision tree to control the microgrid on a given day. We define a performance indicator $Err(t)$ to assess the error in terms of cost between the decisions taken by the EMS algorithm during a testing day and the optimal cost computed by a Q-Learning algorithm when the day is ended.

A. Training phase result

Say, we want to learn the policy that will be used to control the microgrid on the next Tuesday. We use the data collected on the last l previous Tuesdays and train the Q-learning to obtain a decision tree as explained above. To that end, we first train the Q-learning with the Tuesday which is the furthest in time. Then, we train on the following one and so on, until the most recent Tuesday. For each Tuesday, we begin training with the Q-table resulting from the previous Tuesday to accumulate information. For each Tuesday, we use a decreasing number of episodes (ranging from 200 iterations to 30). Regarding the parameters of Q-Learning, we use a separate α for each state-action pair, with an initial value $\alpha(s,a) = 0.2, \forall(s,a)$. Its value decreases with the number of visits to this pair. We set $\gamma = 0.5$, and $\epsilon = 0.99$. ϵ is also decreasing along time because as time passes, and learning improves, less and less exploration of the action space is needed. Performance are best when $l = 4$.

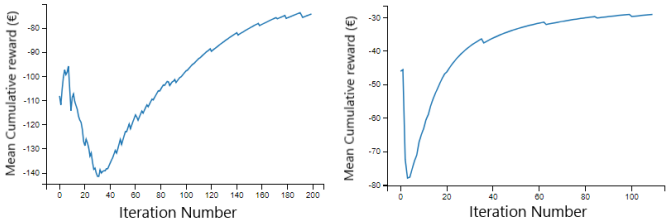

Fig. 2. Mean Cumulated reward for the 1st day (left) and 4th day (right).

Fig. 2 illustrates the improvement in performance of the policy learned on the first Tuesday and the $l = 4$-th one. It shows the "mean cumulative reward" along the episodes. We observe that the performance initially decreases because the Q-Learning explores its environment. At some point, it has learned enough to improve its performance. On the last (l) day used for training, we note that the agent needs less exploration to learn. The algorithm obtains every day an optimal or a near one J_{obj}. The time for computing the entire training phase is 2 seconds on a standard laptop. The size of the Q-table acquired is 74 KB.

B. Testing phase results

To test the algorithm performance, we compute each day, measure the error, and accumulate it over the 52 weeks (n=52). This provides the global error Err_{total} in Eq. (9). The study is run one hundred times in order to measure the variability in performance. The mean of the cost obtained by controlling the EMS is equal to 6491.8€, over the 52 weeks. The optimal cost computed with a Q-Learning algorithm at each testing day is equal to 6452.50€. That gives an averaged Err_{total} of 39.3€, namely 0.6% with a standard deviation of 1.90€.

$$Err_{total} = \sum_{n=1}^{52} Err(t) \tag{9}$$

In order to validate the choice of using a decision tree, different methods were considered. Fig. 3 presents the Err_{total} for each method during only one attempt over n = 52 weeks (which is enough to validate a method).

Finally, we asked a human to perform the same task in order to comparet the performance of our proposed algorithm. The

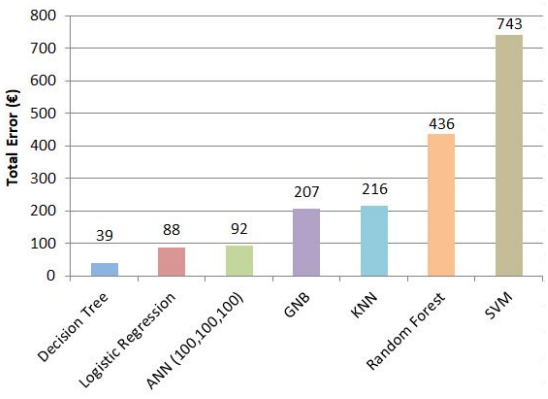

Fig. 3. Benchmark Approximation Methods to the case study.

human gets a Err_{total} of 10 €, outperforming our algorithm but requires 30 minutes of work.

VI. Perspectives and Conclusion

A combination of reinforcement learning and decision tree has been proposed in order to train an agent to manage the power dispatch of a microgrid simulated over 52 weekdays. The approach does not require a forecasting model to predict the future. The last four weekdays are enough to train the RL agent. We saw promising results with only 0.6% of error between the actions take by the EMS agent and the optimal calculated. To improve the agent performance, a generalization method allows to deal adequately with yet unseen states. This is done by introducing a decision tree from what has been learned by Q-Leaning. Based on these first very encouraging results, there are many tracks of research to follow in the future. A more realistic microgrid must be considered with continuous state-action spaces. Furthermore, the study could be improve by decreasing the temporal resolution, from hours to minutes. And finally, multi-agent system could be interesting to study in order to adding a better plug and play ability to the energy management system.

Acknowledgment

This work is funded by the Energy Management Systems department of Total S.A. This work is supported by the TREND-X research program of Ecole polytechnique and the Siebel Energy Institute. The measurements from the SIRTA's (http://sirta.ipsl.fr) outdoors photovoltaic test bench are done in a joint collaboration of 4 laboratories (GeePs, LMD, Limsi and LPICM). Philippe Preux acknowledges the support of Inria and the members of SequeL for the very friendly, fruitful, stimulating, and exciting atmosphere of the group.

References

[1] S. Chu *et al.* Opportunities and challenges for a sustainable energy future. *nature*, 488:294–303, 2012.

[2] Microgrid Knowledge. The Evolution of Distributed Energy Resources. 2018.

[3] R. Lasseter *et al.* Consortium for Electric Reliability Technology Solutions White Paper on Integration of Distributed Energy Resources The CERTS MicroGrid Concept. 2002.

[4] D. Olivares *et al.* Trends in Microgrid Control. *IEEE Transactions on Smart Grid*, 2014.

[5] R.H. Lasseter. MicroGrids. In *IEEE Power Engineering Society Winter Meeting*, volume 1, pages 305–308. IEEE, 2002.

[6] N. Hatziargyriou *et al.* Microgrids. *IEEE Power and Energy Magazine*, 5(4):78–94, 2007.

[7] Farid Katiraei *et al.* Microgrids Management: Controls & Operation Aspects of Microgrids. (june):54–65, 2008.

[8] A.G. Barto R.S. Sutton. *Reinforcement Learning: An Introduction*. Bradford Books, The MIT Press, second edition, 2018.

[9] H. Wang *et al.* Reinforcement Learning in Energy Trading Game among Smart Microgrids. *IEEE Transactions on Industrial Electronics*, pages 1–1, 2016.

[10] A.L. Dimeas and N.D. Hatziargyriou. Agent based Control for Microgrids. In *2007 IEEE Power Engineering Society General Meeting*, pages 1–5. IEEE, 6 2007.

[11] E. Jasmin *et al.* Reinforcement Learning approaches to Economic Dispatch problem. *Electrical Power and Energy Systems*, 2011.

[12] E. Kuznetsova *et al.* Reinforcement learning for microgrid energy management. *Energy*, 59:133–146, 2013.

[13] G.K. Venayagamoorthy *et al.* Dynamic Energy Management System for a Smart Microgrid. *IEEE Transactions on Neural Networks and Learning Systems*, 27(8):1643–1656, 2016.

[14] P. Kofinas *et al.* Energy Management in Solar Microgrid via Reinforcement Learning. In *Proceedings of the 9th Hellenic Conference on Artificial Intelligence - SETN '16*, pages 1–7, New York, New York, USA, 2016. ACM Press.

[15] B. Mbuwir *et al.* Battery Energy Management in a Microgrid Using Batch Reinforcement Learning. *Energies*, 10(11):1846, 11 2017.

[16] P. Kofinas *et al.* Fuzzy Q-Learning for multi-agent decentralized energy management in microgrids. *Applied Energy*, 219:53–67, 6 2018.

[17] Migan *et al.* Multi-technology photovoltaic module test bench on the sirta meteorological and climate observatory. 2015.

[18] Charlie Furrer. The Economics of the Tesla Powerwall 2, 2016.

[19] R. Bellman. *Dynamic programming*. Princeton University Press, 2010.

[20] L. Breiman *et al.* *Classification and Regression Trees*. The Wadsworth and Brooks-Cole statistics-probability series. Taylor & Francis, 1984.