Energy Management for Microgrids: a Reinforcement Learning Approach

Tanguy Levent, Philippe Preux, Erwan Le Pennec, Jordi Badosa, Gonzague Henri, Yvan Bonnassieux

Energy Management for Microgrids: a Reinforcement Learning Approach

Tanguy Levent
PICM Laboratory
Ecole Polytechnique
Paris, France
tanguy.levent@polytechnique.edu

Gonzague Henri
PICM Laboratory
Ecole Polytechnique
Paris, France
yvan.bonnassieux@polytechnique.edu

Philippe Preux
CRIStAL
Universit de Lille
Villeneuve d'Ascq, France
philippe.preux@inria.fr

Jordi Badosa
PICM Laboratory
Ecole Polytechnique
Paris, France
yvan.bonnassieux@polytechnique.edu

Erwan Le Pennec
PICM Laboratory
Ecole Polytechnique
Paris, France
yvan.bonnassieux@polytechnique.edu

Yvan Bonnassieux
PICM Laboratory
Ecole Polytechnique
Paris, France
yvan.bonnassieux@polytechnique.edu

***Abstract*—This paper presents a reinforcement learning based framework for energy management and economic dispatch in an islanded microgrid without any forecasting module. The architecture of the algorithm is divided in two parts: a learning phase trained by an optimized reinforcement learning algorithm with a past and small dataset and the execution phase with a dynamic decision tree created by the first step. One advantage of this approach is to create an autonomous agent, able to react in real-time, considering only the past, thus no forecasting algorithm is needed. This framework was tested on real data acquired at Ecole Polytechnique in France over a long period, 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, Supervised Learning, Reinforcement Learning**

I. Introduction

For the electricity sector to adapt to climate change, and be part of the solution, the sector will have to evolve toward a smart and decarbonize electricity 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. Researched 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, microgrid have been pushed as one of the possible solution [2]. A microgrid is a single controlled entity and has several advantages: maintaining the balance of the utility grid, reducing the peak, reducing periods of load variability, enhancing the power quality and reliability services and decrease the feeder loses. Nevertheless, many challenges must be overcome to deploy microgrids at scale. Among those challenges, 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 managenement optimization, the tertiary control level of the hierarchical architecture, as proposed in [3]–[5] is considered.

In the past few years, machine learning (ML), a branch of artificial intelligence, became very popular and shows promises in a number of areas [citation]. Reinforcement Learning (RL) is a branch of ML with the aim to develop agents that can interact with an environment, and learn form it. It is used for decision making under uncertainty and performs well for sequential problems [6]–[8]. Reinforcement Learning has already been proposed for microgrid in energy trading market [9], distributed generation (DG) agents learning [10], energy management [11]–[16]. This paper proposes a new RL approach with a new term called "Q-transfer", explained in Section IV, for economic dispatch problem combined with a supervised learning algorithm to select the best policy in every situation. This specific architecture algorithm has not been studied and its advantages are: no large set of data is needed, no forecast is needed, online and model-free algorithm, very fast, almost in real-time, adaptive and scalable.

The paper is organized as follows: In Section II, the microgrid case study is presented and the objective function, constraints and hypothesis are defined. The Section III describes the architecture of this new approach/framework. In the Section IV, we explain the reinforcement method followed by the Section V, where we describes the decision tree algorithm used to execute what the agent learns. Section VI presents the results of the application proposed and finally a perspective and conclusion are exposed in the Section VII.

II. Design and Modeling of the Microgrid

The microgrid design for this study is simple in its architecture and in the number of DERs used but makes sense to

IEEE ISGT Conference 2019, Bucharest, Romania

test the performance of the algorithm. The modeling and the simulation is built on real databases provided by the TREND-X research program of Ecole Polytechnique and from SIRTA (http:/sirta.ipsl.fr), an Atmospheric Research Observatory, both based in Palaiseau, France at 200 meters from each other. It necessary to understand this paper as the first stone based on a top-down logical approach for future works.

A. Microgrid and Units

The model is a remote microgrid which is not connected to the distribution grid. It consists of solar renewable sources only (PV), a set of batteries, a diesel generator (genset) and loads. The different units used to the case study are defined as (figure 1):

- **PV**: polycristallin solar pannels produce by Francewatts company with a total power capacity of 15 kW_p. The panels come from the SIRTA observatory, they use an MTTP.
- **Genset**: diesel generator Winco DR20I4 of 20 kW and artificially create for the study
- **Batteries**: Tesla powerwall 2 of 60 kWh of usable capacity and artificially create for the study
- **Loads**: Real data come from the Ecole Polytechnique Innovation building. The electricity consumption depicts the start-up, workers and student usages.

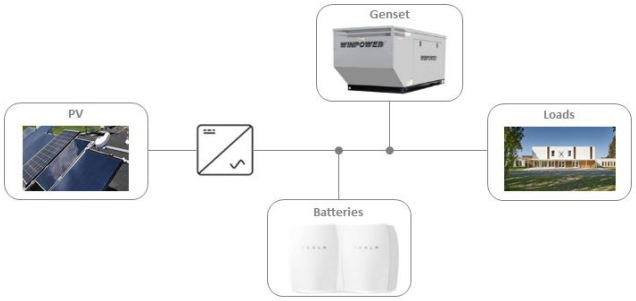

Fig. 1. Structure of the studied Microgrid

B. Problem Statement

The paper wants to answer the economic dispatch problem for the microgrid case study. It means minimizing the operational costs of the entire system while meet the demands in proper time. The units describes in the previous section must be managed by an supervisory entity called the Energy Management System (EMS), with the intention of doing the optimal instructions to save operation costs. The Figure 3 depicts the value of having a good algorithm inside the EMS to handle the economic dispatch problem. As the algorithm doesn't works with a forecast to then define a scheduling of a day ahead, each decision will be done directly regarding the information given by the microgrid environment. For this reason, this study doesn't fall in the Unit Commitment problem but both respect the same rules like keeping the power balance between the supply and the demand.

C. Objective function

The objective function of the Microgrid EMS seeks to minimize the power system costs into the operations over a period of time (T) and it is formulated as $\min J_{cost}$ where:

$$J_{cost} = \sum_{t=0}^{T} |P_B(t)| * C_B(t) + P_G(t) * C_G(t) + P_C(t) * C_C(t). \quad (1)$$

P_B and C_B represent the charge/discharge power (reason why P_B is write in absolute value) and the cost associated of the set of batteries. P_G and C_G are the power output and the cost associated of the diesel generator. Finally P_C and C_C are the power and the cost during a crash because of disturbance due to bad management. The time step is defined by t. The study does not consider a cost for the solar panels because it is included in the system and the unit does not need smart operation management to optimize its cost (contrary to batteries). Finally this study is not concerned by demand response so the loads are not subject to shedding or shifting.

D. Constraints and Hypothesis

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

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

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

$$P_{Bcap}(t) = P_{Bcap}(t-1) - P_B(t) \quad (3)$$

with positive value for P_B if the battery discharges in the microgrid and a negative one if it charge power from the microgrid. It is not possible for the battery to be in a charge and discharge modes during the same time step. It is described as:

$$P_B^+(t) + P_B^-(t) \leq 1 \quad (4)$$

where P_B^+ and P_B^- are binary variables. Furthermore, the battery capacity must stay within its limits at any time:

$$P_{Bmin} \leq P_{Bcap}(t) \leq P_{Bmax}. \quad (5)$$

The battery cost in €/kWh is related to the number of cycles during its lifespan. The duration life of a battery is highly affected by how the user exploits it in terms of charge and discharge. The more the battery discharges deeply relative to the overall capacity of the battery, the shorter its lifespan: this notion is known as the "Depth of Discharge" (DoD). For the study, we have considered an approximate fixed cost in €/kWh, denoted m, according to [17].

$$C_B(t) = m. \quad (6)$$

The diesel generator is also a simplified model because the fuel consumption is usually represented as a non linear function. In this study, we consider it as a linear function, thereby the cost C_G is fixed, represented by the variable q and only the delivered power is responsible for the diesel generator cost, expressed by:

$$C_G(t) = q. \tag{7}$$

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

$$0 \leq P_G(t) \leq P_{Gmax} \tag{8}$$

knowing that if $P_G(t)$ is equal to zero, the genset is considered in a turn off mode.

Finally the study managed a net demand P_{Net}, which is the difference between the power output of the solar panels and the consumption of the school building. The demand is feed first with the power production of renewable. The excess or deficit following at each step is the amount of power to supervise:

$$P_{Net}(t) = P_{PV}(t) - P_L(t) \tag{9}$$

III. Reinforcement Learning to control an EMS

Reinforcement learning (RL) [18] is based on the idea of learning to act in an optimal way by interacting with an environment. To be more specific, let us assume a learning agent (algorithm) that behaves sequentially along time t. At each t, the agent perceives the state of its environment $s_t \in \mathcal{S}$, decides on an action $a_t \in \mathcal{A}$ to perform, emits it. 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 fullfill a task, that is optimize a certain objective function: the most common such objective function is $\sum_{t\geq 0} \gamma^t r_t$, with $\gamma \in]0,1[$: maximizing this objective 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 curent actions. The environment is usually 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 environment is usually assumed stationary.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. Partial observability means that the assumption that the agent has access to the true state of the environment can not be met here. Then, it 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.

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 (optimally) a microgrid. For this purpose, we define the different elements of this tuple.

A. States

The state is the lens through which the agent perceives its environment. The state design is decisive in the performance. It 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 enough to describe the microgrid. It is important to keep in mind that an action influences the next steps. If the agent decision is to discharge, the battery capacity will be impacted. An hourly action bring automatically the agent in the next hour for a new decision.

B. Actions

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

- action 1 = Charge: batteries charge electricity.
- action 2 = Discharge: batteries discharge electricity.
- action 3 = Genset: genset is turned on and produces electricity.
- action 4 = Do nothing.

C. Reward function

The return characterizes the achievement of the task and also constraints. It is represented as a real value. In the study the current reward is associated with the cost of the unit which is used to manage the net demand P_{Net}. Each unit has its own cost and we also consider a cost if the constraints are not respected. This cost does not represent a real cost because it is complicated to estimate the damage due to a crash but it is forced at a very low level in order to advise the agent of a bad choice in the action selection in a certain situation. The Equation 11 below represents the reward conditional function:

$$r(s,a) = \begin{cases} m, & \text{if charge or discharge the battery} \\ q, & \text{if power produced by the genset} \\ 0, & \text{if do nothing if well called} \\ c, & \text{if the constraints are not respected} \end{cases} \tag{10}$$

D. Q-Learning: a reinforcement learning algorithm

Q-learning is an algorithm to solve an RL problem, that is compute its optimal policy. A policy assigns a probability distribution to each state on the set of actions. In the case of MDPs 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). We denote this policy π, $\pi(s)$ being this best action for state s. A key concept is the "value" of a state-action pair which formalizes how good it is for the agent to emit a certain action a be in a state s, with regards to the optimization of the objective function, that is the fullfillment 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. 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:

$$\begin{aligned} Q_{t+1}(s,a) =& Q_t(s,a) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1},a) \\ & - Q_t(s,a)] \end{aligned} \tag{11}$$

The series Q_t converges under suitable assumptions to Q^*. This equation is the essence of the Q-Learning algorithm.

Q is a function and has to be represented somehow. 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. (11). Gradually, the agent will learn 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.

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 (11)
    end while
end for
```

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 this problem. The one chosen is called the "ϵ-greedy method" (see line 6 of algorithm 1). After enough iterations, the learning agent converges toward a good policy.

E. Q-transfer: a new way of optimizing the algorithm

The agent must learn over the last 4 days, selected by the execution day that the EMS will manage. To speed up the convergence a new approach is presented here as Q-transfer. Instead of learning the four days directly, the method suggests to split the learning phase by day. Each day improve the learning of the next one. The principle is the agent transfers his "day 1 Q-table knowledge" trained to initialize a new Q-table for the next 24 hours: day 2. Thereby, the agent divide his way of learning. Instead of tackle directly the problem of dispatch power over 96 hours, the study proposes to address the problem by a one day training, understand it very well by playing several episodes and then pass what the agent learns for the new training day. And so on and so forth until the fourth day. It was proved that the ability decision making of the agent has robustly enhanced. The Figure 7 depicts the mean cost evolution over number of iterations between a Q-transfer approach and a classical approach. The convergence is speed up by 10 times with Q-transfer.

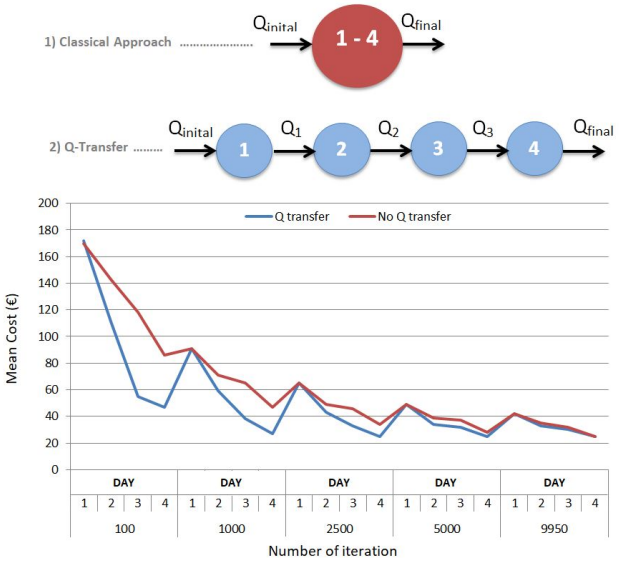

Fig. 2. Benchmark speed convergence between Q-transfer and classical approach

IV. Decision Tree, the performer

At the end of the learning phase, a Q-Knowledge or Q_{Final} table is transmitted to the execution phase in order to extract the understanding gained with the Q-Learning algorithm. The maximum state-action value for each state is store in a new memory variable to keep only the state and the best action associated. This new data - corresponding of the best agent interactions into the simulated microgrid - feeds a decision tree algorithm. The aim is to provide the ability to generalize some experiences during the training phase into general rules. These rules will be crucial to decide the action for each time step during the execution day. The advantage of generalize is to give the possibility at the agent to make a good decision even if he never meets a situation before.

A decision tree is a non-parametric supervised learning algorithm for regression and classification. 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 data and a prediction model is made at each division. The decision tree looks like a flowchart diagram where the terminal nodes represents the target decisions (the agent action), and the internal nodes the attributes (the environment states) with thresholds. There are different algorithms (ID3, C4.5, C5.0, ...) for constructing a tree, but the one used for the study is CART (Classification and Regression Trees) [20]. The latter is a binary tree construction, means at each node only two branches are created. Once the Q-Knowledge feeds the CART decision tree algorithm, decision rules describes by a flowchart are created as depicted in the Figure 8.

V. Results of the study

Regarding the performance of the global algorithm, it is important to split it into two parts. One section for validate the training phase performance with Q-Learning algorithm and another one for the testing phase, which is the main result to answer the objective function write of the study. The study will be done over one random weekday during one year to validate the interest of the algorithm. Thus, the period of time (T) is equal to 52 weeks, beginning the 07/15/2016 . After run through an execution day, a Q-Learning algorithm is still performs to calculate the optimal cost. The difference between the cost obtains with the execution phase (decision tree) and the latter will be the cost performance error, note Err. The error will be cumulated over T. Higher is the error, lower is the algorithm interest to respond at the objective function. The result will be compared with a human decision making. Below is describes the algorithm performance error:

$$Err_{total} = \sum_{t=0}^{51} Err(t) \tag{12}$$

A. Training phase result

First of all, the Q-Learning algorithm must to be implemented with manually hyperparameters. For the study, it is decided to use $\alpha = 0.2$ decreasing with pair state-action encountered, $\gamma = 0.5$ and $\epsilon = 0.99$ decreasing with time. An episode go on 24 hours, namely 24 time steps. The number of episode starts at 200 for the first day and decreases by 30 for the following days. Recall that the Q-Learning is performs over the past 4 weekdays. The Q-transfer is achieved between each day. To measure the performance of the algorithm, the cost function is calculated and the optimal one as well. A good Q-Learning algorithm is necessary to produces a good decision tree for the execution day. In order to follow the agent learning development, the curve in the Figure 9 and 10 are drawn.

It describes the "mean cumulative reward" over the number of episode. We observe that during the first part, the curve decreases because the agent wants to explore his environment. At a moment the agent understood and learned from it,

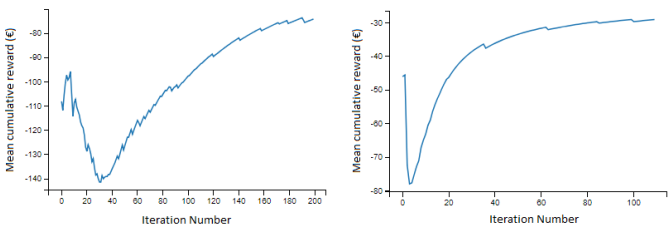

Fig. 3. Cumulated reward for the 1st day (left) and 4th day (right)

involving an enhanced decision making and increasing the mean cumulative reward until the end. Notice that the agent needs less exploration at the fourth day to learns from the environment compared to the 1st day because of the Q-transfer. The training phase algorithm obtains an **optimal** result over each days. It means that the algorithm performs very well and converges every time to the global optimum. The time for computing the entire training phase is equal to **2 seconds**. The size of the Q-Knowledge table acquired is equal to **73.824 KB**.

B. Testing phase results

Once the Q-Knowledge table is obtained, a CART decision tree algorithm is implemented. The latter will be conserved without any modification throughout the execution day. To test the algorithm performance, the global error Err_{total} will be calculated at the end of the 52nd week. The figure 11 describes this process:

The study is run over one hundred time in order to observe the standard deviation between each attempt. The mean of the J_{cost} obtained by training the EMS with an optimized Q-Learning and a CART decision tree is equal to **6491.8 €**. The Optimal cost calculated with an Q-Learning algorithm at each execution day is equal to **6452.50 €**. That gives us in average an Err_{total} of **39.3 €** for 52 weekdays energy management with an average standard deviation of **1.90€**.

In order to validate the decision tree algorithm, the study have been tested over different approximation methods. The barchart below presents the Err_{total} for each method during only one attempt (which is enough to validate a method).

Finally, the case study have been perform by a human to compare the performance between them. It shows that the human beat the algorithm but with only few savings and a lot of time.

VI. Perspectives and Conclusion

An optimized Reinforcement Learning combined with a Decision Tree method have been proposed in this paper in order to learns properly an agent to manage the power dispatch into a microgrid environment simulated during a long period of time. The approach don't requires a forecast model for mapping the future to implement an optimization method to resolve the problem of economic dispatch. Even if the case study model is simple compared to a real microgrid, the new framework presented reveal very good results, with

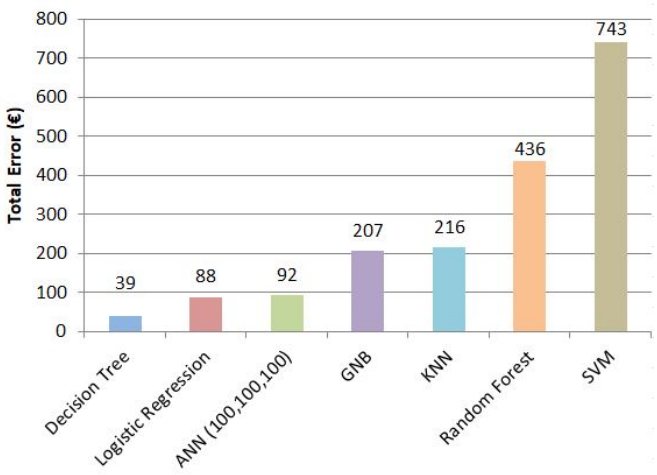

Fig. 4. Benchmark Approximation Methods to the case study

only 0.6% of error and a very short time computation. With only four weeks the agent can understand the net demand distribution of a day, take action over them to then reinforce his knowledge. An approximation method is necessary to generalize the agent behavior regarding what he learns during the training phase in order to act properly during the execution day. Nevertheless, the study muste be enhanced by taking into account realistic simulation of the different microgrid units and brings not deterministic states and actions but continuous ones. Furthermore, the study could be improve by decrease the temporal horizon in minutes. Adding a multi-timescale forecast algorithm into this framework will be beneficial in order to get set-point for instance to the battery level. A next step should be having an algorithm which can acts in islanded and connected mode depending of the distribution level operations. And finally, multi-agent system based on this approach could be interesting to study in order to adding a better plug and play ability to the energy management system.

ACKNOWLEDGMENT

REFERENCES

[1] Steven Chu and Arun Majumdar. Opportunities and challenges for a sustainable energy future. 2012.

[2] Marnay C Stephens J Dagle J Guttromson R Sakis Meliopoulous A Yinger R Eto J Lasseter R, Akhil A. Consortium for Electric Reliability Technology Solutions White Paper on Integration of Distributed Energy Resources The CERTS MicroGrid Concept. 2002.

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

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

[5] Nikos Hatziargyriou Aris Dimeas Farid Katiraei, Reza Iravani. Microgrids Management: Controls & Operation Aspects of Microgrids. (june):54–65, 2008.

[6] David Silver and et al. Mastering the game of Go without human knowledge. 2017.

[7] Volodymyr Mnih and et al. Playing Atari with Deep Reinforcement Learning. Technical report, 2013.

[8] Volodymyr Mnih and et al. Human-level control through deep reinforcement learning. 2015.

[9] Huiwei Wang and 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] Ea Jasmin, TP Imthias Ahamed, and VP Jagathy Raj. Reinforcement Learning approaches to Economic Dispatch problem. *Electrical Power and Energy Systems*, 2011.

[12] Elizaveta Kuznetsova, Yan-Fu Li, Carlos Ruiz, Enrico Zio, Graham Ault, and Keith Bell. Reinforcement learning for microgrid energy management. *Energy*, 59:133–146, 2013.

[13] Ganesh Kumar Venayagamoorthy, Ratnesh K. Sharma, Prajwal K. Gautam, and Afshin Ahmadi. Dynamic Energy Management System for a Smart Microgrid. *IEEE Transactions on Neural Networks and Learning Systems*, 27(8):1643–1656, 2016.

[14] Panagiotis Kofinas, George Vouros, and Anastasios I. Dounis. 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] Brida Mbuwir, Frederik Ruelens, Fred Spiessens, Geert Deconinck, Brida V. Mbuwir, Frederik Ruelens, Fred Spiessens, and Geert Deconinck. Battery Energy Management in a Microgrid Using Batch Reinforcement Learning. *Energies*, 10(11):1846, 11 2017.

[16] P. Kofinas, A.I. Dounis, and G.A. Vouros. Fuzzy Q-Learning for multi-agent decentralized energy management in microgrids. *Applied Energy*, 219:53–67, 6 2018.

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

[18] Andrew G. Barto Richard S. Sutton. *Reinforcement Learning: An Introduction*. Second edition, 2018.

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

[20] L Breiman, J Friedman, C J Stone, and R A Olshen. *Classification and Regression Trees*. The Wadsworth and Brooks-Cole statistics-probability series. Taylor & Francis, 1984.