University of Liège
Faculty of Applied Sciences
Department of Electrical Engineering & Computer Science

PhD dissertation

CONTRIBUTIONS TO DEEP REINFORCEMENT LEARNING AND ITS APPLICATIONS IN SMARTGRIDS

---

by Vincent François-Lavet

Advisors: Damien Ernst & Raphaël Fonteneau

ABSTRACT

Reinforcement learning and its extension with deep learning have led to a field of research called deep reinforcement learning. Applications of that research have recently shown the possibility to solve complex decision-making tasks that were previously believed extremely difficult for a computer. Yet, deep reinforcement learning requires caution and understanding of its inner mechanisms in order to be applied successfully in the different settings.

As an introduction, we provide a general overview of the field of deep reinforcement learning. The thesis is then divided in two parts.

In the first part, we provide an analysis of reinforcement learning in the particular setting of a limited amount of data and in the general context of partial observability. In this setting, we focus on the tradeoff between asymptotic bias (suboptimality with unlimited data) and overfitting (additional suboptimality due to limited data), and theoretically show that while potentially increasing the asymptotic bias, a smaller state representation decreases the risk of overfitting. An original theoretical contribution relies on expressing the quality of a state representation by bounding L_1 error terms of the associated belief states. We also discuss and empirically illustrate the role of other parameters to optimize the bias-overfitting tradeoff: the function approximator (in particular deep learning) and the discount factor. In addition, we investigate the specific case of the discount factor in the deep reinforcement learning setting case where additional data can be gathered through learning.

In the second part of this thesis, we focus on a smartgrids application that falls in the context of a partially observable problem and where a limited amount of data is available (as studied in the first part of the thesis). We consider the case of microgrids featuring photovoltaic panels (PV) associated with both long-term (hydrogen) and short-term (batteries) storage devices. We propose a novel formalization of the problem of building and operating microgrids interacting with their surrounding environment. In the deterministic assumption, we show how to optimally operate and size microgrids using linear programming techniques. We then show how to use deep reinforcement learning to solve the operation of microgrids under uncertainty where, at every time-step, the uncertainty comes from the lack of knowledge about future electricity consumption and weather dependent PV production.

JURY MEMBERS

Louis Wehenkel, Professor at University of Liège (President);

Damien Ernst, Professor at University of Liège (Advisor);

Raphael Fonteneau, Researcher and adjunct assistant lecturer at University of Liège (Co-advisor);

Kurt Driessens, Professor at Maastricht University;

Remi Munos, Research scientist at DeepMind and senior researcher at Inria;

Olivier Teytaud, Research scientist at Google Research and senior researcher at Inria;

ACKNOWLEDGMENTS

I'm grateful to all the people who have given me encouragements, advices and showed me their support in various ways.

First of all, I would like to thank my two supervisors Damien Ernst and Raphael Fonteneau for the helpful discussions during this thesis. I thank them for their support and the freedom they gave me to pursue my research.

I also want to thank all the members of the jury for their interest in my thesis and for taking the time to evaluate it.

I would like to thank my colleagues who all contributed to a great working environment that was both productive and enjoyable. I'd like to warmly thank all the colleagues from the Montefiore Institute and from the smartgrids team. In particular, I would like to thank my office mates: Aaron, Arnaud and Mathilde who have shared quite a lot of my time while I was working on this thesis and who have all contributed in their way to this thesis with advices and encouragements. I also would like to particularly thank all the other people that I have had the chance to work and closely interact with: Quentin, Samy, David, Michael, Adrien, Efthymios, Frederic, Antonio, Gilles, Pierre, Grégoire.

I would like to thank Danielle Bonten, Sophie Cimino, Cécile Duchesne and Diane Zander who have been of efficient and kind help for different administrative procedures.

I also want to thank the Consortium des Equipements 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 part of this research. And I specially thank David Collignon and Frédéric Wautelet for their helpfulness.

Finally, I want to warmly thank my family and friends who supported me in both good and bad times. Last but not least, I also want to specially thank Anne-Valentine for sharing with me these almost four years during my PhD.

CONTENTS

INTRODUCTION AND OVERVIEW OF DEEP REINFORCEMENT LEARNING

1 INTRODUCTION

A robot willing to open a door, a drone avoiding obstacles, a trader managing portfolios, a dog trying to catch a ball, a chess player considering the next moves, a teenager playing a computer game, a man willing to paint a wall: in all of the previous cases, certain sequences of actions will allow some objectives to be achieved. To that end, every action has an influence on the next situation, which possibly has short- and/or long-term effects. In addition, the environment is usually not entirely predictable and it is not possible to consider only one sequence of actions for a given task. Instead the agent has to be able to act under uncertainty and be ready for any possible situation.

A whole field of research is dedicated to learn how to act in an environment from past experiences. This field is named reinforcement learning and is inspired by behaviorist psychology. The idea is that an artificial agent, similarly to a biological agent, may learn by interacting with its environment. With experience gathered, the artificial agent should be able to optimize some objectives. This approach is extremely general and applies in principle to any type of sequential decision-making problem relying on past experience to take decisions. The environment may be stochastic, the agent may only observe partial information about the current state, the observations may be high-dimensional (e.g., frames and time series), the agent may freely gather experience in the environment or, on the contrary, the amount of data may be fixed or limited due to various reasons, etc.

Over the past few years, reinforcement learning has become more and more popular in addressing the most challenging sequential decision-making problems. This happened mainly thanks to the extension of reinforcement learning with deep learning which allows learning different levels of abstractions from data (such as those necessary to perform machine vision or natural language processing tasks). The resulting field, known as deep reinforcement learning, opens up the perspective to mimic some cognitive functions of humans such as problem solving even in high-dimensional space—which, only a few years ago, was difficult to conceive due to the curse of dimensionality.

Thanks to these developments, new automatic decision-making processes are likely to impact a wide variety of fields such as finance, marketing, clinical trials, robotics, self-driving, smartgrids, etc.

USING DEEP REINFORCEMENT LEARNING IN SMARTGRIDS

In particular, in the domain of smartgrids, deep reinforcement learning algorithms open up the perspectives of significantly improving current control schemes to solve decision and control problems associated with electric power systems. Those improvements are particularly important in the current context where the power system landscape evolves and faces new challenges with the integration of distributed renewable electricity generation capacities (e.g., photovoltaic panels and wind turbines).

However, operation problems in the domain of smartgrids exhibit in many cases the specificities of the most challenging problems of sequential decision-making problems under uncertainty:

1. The problems are partially observable: the future evolution of the system cannot be predicted accurately based solely on the current observation as the agent has only access to some observations of an actual hidden phenomenon. In this setting, defining a policy based solely on the current observation will likely lead to suboptimal solutions. It is therefore often important to look at a history of past observations but the longer the history, the more complex is the task and specific tradeoffs have to be made to build a good operation.

2. Available data is constrained: the agent does not have the possibility to freely gather data due to two reasons. First, a cost concern prevents making a large number of trials and errors in the actual environment because this leads to suboptimal or even unsafe operations (causing damage to the system). Second, a time constraint requires to obtain a policy without having the possibility to gather data for many years, thus limiting the information about the environment (e.g., on the natural phenomena involved).

METHODOLOGICAL WORK IN THIS THESIS

In the partially observable case, few contributions in the field actually provide clear and insightful explanations about some very general algorithmic design choices, such as the impact of a particular choice of the state representation (features chosen to build the policy). The lack of theoretical understanding is particularly striking in the context where the available data is constrained and that it can not be phased out by gathering more data using strategies balancing the exploration/exploitation (E/E) tradeoff. Following these considerations, the main goals of this thesis are to provide (i) an in-depth analysis of the effect of limited data in reinforcement learning with a focus on the partially observable setting, and (ii) investigate a concrete real-

world application that makes use of this analysis in the domain of smartgrids.

1.1 OUTLOOK AND CONTRIBUTIONS

First, an introduction to the deep reinforcement learning field is presented in Chapter 2. The thesis is then divided into two main parts: (i) theoretical/methodological contributions to (deep) reinforcement learning and (ii) the smartgrids application.

With regard to the theoretical and methodological aspects, we provide in chapter 3 an analysis in a particular setting where a limited amount of data is available and in the general context of partial observability. In this setting, a tradeoff between asymptotic bias (suboptimality with unlimited data) and overfitting (additional suboptimality due to limited data) is formalized. It is also theoretically shown that a smaller state representation decreases the risk of overfitting but potentially increases the asymptotic bias. This original analysis relies on expressing the quality of a state representation by bounding L_1 error terms of the associated belief states. Theoretical results are empirically illustrated when the state representation is a truncated history of observations. We also discuss and empirically illustrate how using function approximators and adapting the discount factor may enhance the tradeoff between asymptotic bias and overfitting. In Chapter 4, the discussion is then extended to the role that the discount factor may play in the quality of the learning process of a deep Q-network (DQN) in the online setting case. When the discount factor progressively increases up to its final value while gathering new data, it is empirically shown that it is possible to significantly reduce the number of learning steps. This phenomenon is related to the bias-overfitting tradeoff as well as the instabilities of neural networks when used in a value-based reinforcement learning scheme.

In the second part of this thesis we focus on an application in the domain of smartgrids. The case of a microgrid featuring photovoltaic panels (PV) associated with both long-term (hydrogen) and short-term (batteries) storage devices is considered. In Chapter 5, a formalization of the problem of building and operating microgrids interacting with their surrounding environment is proposed. In the context where the consumption and the production are known, how to optimally operate a microgrid using linear programming techniques is studied. It appears that this optimization technique can also be used to address the problem of optimal sizing of the microgrid. In Chapter 6, we consider the question of activating the storage devices in the more realistic context of a sequential decision-making problem under uncertainty where, at every time-step, the uncertainty stems from the lack of knowledge about future electricity consumption and weather dependent PV production. This problem is addressed using deep reinforcement learning. Making use of the

methodological aspects developed in the first part of this thesis, a specific deep reinforcement learning approach is designed in order to extract knowledge from past consumption and production time series, as well as any available forecasts.

1.2 PUBLICATIONS

This dissertation is based on different contributions in the domain of deep reinforcement learning and smartgrids:

- *On overfitting and asymptotic bias in batch reinforcement learning with partial observability*, Vincent François-Lavet, Damien Ernst and Raphael Fonteneau, to be published.
- [François-Lavet et al., 2015] *How to Discount Deep Reinforcement Learning: Towards New Dynamic Strategies*, Vincent François-Lavet, Raphael Fonteneau, and Damien Ernst. In NIPS 2015 Workshop on Deep Reinforcement Learning.
- [François-Lavet et al., 2016a] *Towards the Minimization of the Levelized Energy Costs of Microgrids using both Long-term and Short-term Storage Devices*, Vincent François-Lavet, Quentin Gemine, Damien Ernst and Raphael Fonteneau. In Smart Grid: Networking, Data Management, and Business Models, 295–319, 2016, CRC Press.
- [François-Lavet et al., 2016b] *Deep Reinforcement Learning Solutions for Energy Microgrids Management*, Vincent François-Lavet, David Taralla, Damien Ernst and Raphael Fonteneau. In European Workshop on Reinforcement Learning, 2016.

During this thesis, other collaborations have also led to effective contributions, even though they are not discussed within this thesis (except the first one that is provided in Appendix A):

- [François-Lavet et al., 2014] *Using approximate dynamic programming for estimating the revenues of a hydrogen-based high-capacity storage device*, Vincent François-Lavet, Raphael Fonteneau, and Damien Ernst. In IEEE International Symposium on Adaptive Dynamic Programming and reinforcement Learning (ADPRL 2014), Orlando.
- [Sutera et al., 2014] *Simple Connectome Inference from Partial Correlation Statistics in Calcium Imaging*, Antonio Sutera, Arnaud Joly, Vincent François-Lavet, Zixiao Aaron Qiu, Gilles Louppe, Damien Ernst, and Pierre Geurts. In Neural Connectomics. 2014, 23–34.
- [Grégoire et al., 2015] *Electricity storage with liquid fuels in a zone powered by 100% variable renewables*, Grégoire Leonard,

Vincent François-Lavet, Damien Ernst, Christoph J Meinrenken and Klaus S Lackner. In European Energy Market (EEM), 2015 12th International Conference on the European Energy Market (EEM), 1–5, 2015, IEEE.

- [Aittahar et al., 2015] *Imitative Learning for Online Planning in Microgrids*, Samy Aittahar, Vincent François-Lavet, Stefan Lodeweyckx, Damien Ernst and Raphael Fonteneau. In International Workshop on Data Analytics for Renewable Energy Integration, 1–15, 2015, Springer.

- [Castronovo et al., 2017] *Approximate Bayes Optimal Policy Search using Neural Networks*, Castronovo, Michaël and François-Lavet, Vincent and Fonteneau, Raphaël and Ernst, Damien and Couëtoux, Adrien. In 9th International Conference on Agents and Artificial Intelligence (ICAART 2017).

The work related to the business cases of microgrids [François-Lavet et al., 2016a] has also been featured in the Belgian business newspaper "l'Echo" for which we have provided specific simulations:

- "Même sans compensation, les panneaux photovoltaïques sont rentables", Christine Scharff, L'Echo, Nov, 2015.

This thesis has also motivated the development of the open-source project DeeR (that stands for "deep reinforcement"), which served as the basis of the work presented in Chapter 6:

`http://deer.readthedocs.io`

2 OVERVIEW OF DEEP REINFORCEMENT LEARNING

2.1 OUTLINE

In this chapter, the field of machine learning and the deep learning approach are first introduced. The goal of that section is to provide the general context of machine learning and explain briefly how and why deep learning has a profound impact on the whole field of machine learning. In the following of the chapter, the focus is on a particular subfield of machine learning, that is (deep) reinforcement learning (RL). The different components that an RL agent can use to learn a task are examined. After this, the challenges in RL are presented. The different settings of tasks that can be learned via RL are also reviewed. Finally, how deep RL can be used to solve many different types of real-world problems are discussed.

This chapter is not intended to be a self-contained overview but it aims to introduce the main elements of deep RL and guide the reader towards appropriate resources for the details.

2.2 GENERAL INTRODUCTION TO MACHINE LEARNING AND DEEP LEARNING

Machine learning relates to the capability of computers to learn from examples or interactions with an environment without following explicitly defined rules. Three types of machine learning tasks can be described:

- Supervised learning is the task of inferring a classification or regression from labeled training data.
- Unsupervised learning is the task used to draw inferences from datasets consisting of input data without labeled responses.
- Reinforcement learning (RL) is the task concerned with how software agents ought to take actions in an environment in order to maximize cumulative (delayed) rewards.

Mainly thanks to the recent development of deep learning, these three types of tasks have undergone dramatic improvements when working with high-dimensional data such as time series, images and videos.

The resurgence of interest in deep learning mainly comes from the following three aspects (which are complementary and lead to a virtuous circle): (i) an exponential increase of computational power

(with the use of GPUs), (ii) methodological breakthroughs in deep learning (such as [Srivastava et al., 2014; Ioffe and Szegedy, 2015; He et al., 2016; Szegedy et al., 2016; Klambauer et al., 2017]) and (iii) a growing eco-system of softwares and datasets.

Deep learning finds its origin in the attempt to model the neural processing in the brain of biological entities. Subsequently deep learning has become incompatible with current knowledge of neurobiology [Bengio et al., 2015] but there exists nonetheless some parallels such as the convolutional layers that are inspired by the organization of the animal visual cortex [Fukushima and Miyake, 1982; LeCun et al., 1998].

In its most abstract form, an artificial neural network (or simply neural network) is a function $f : \mathcal{X} \to \mathcal{Y}$ parameterized with $\theta \in \mathbb{R}^{n_\theta}$ that takes as input $x \in \mathcal{X}$ and gives as output $y \in \mathcal{Y}$ ($\mathcal{X}$ and $\mathcal{Y}$ depend on the application):

$$y = f(x; \theta) \tag{2.1}$$

The specificity of deep neural networks is their structures made of multiple processing layers composed of non-linear transformations. Within one given neural network, an arbitrarily large number of layers is possible, and the trend in the last few years is to have an ever-growing number of layers (>100 in supervised learning tasks).

In general, the complexity of the function approximator provides upper bounds on the generalization error, which is defined empirically as the difference between the training and test errors (see Figure 2.1 for an illustration). The generalization error can be bounded by making use of complexity measures, such as the Rademacher complexity [Bartlett and Mendelson, 2002], or the VC-dimension [Vapnik, 1998]). However, even though it lacks strong theoretical foundations, it has become clear in practice that the strength of deep neural networks is their generalization capabilities, even with a high number of parameters n_θ (hence a potentially high complexity) [Zhang et al., 2016].

The simplest type of neural network is the one made entirely from fully-connected layers. We will briefly describe such a neural network with one hidden layer (see Fig 2.2). The first layer is given the input values (i.e., the input features) x in the form of a column vector $[n_x]$ ($n_x \in \mathbb{N}$). The values of the next hidden layer are a transformation of these values by a non-linear parametric function, which is a matrix multiplication by $W_1[n_h \times n_x]$ ($n_h \in \mathbb{N}$), plus a bias term $b_1[n_h]$, followed by a non-linear transformation such as ReLu:

$$h = \mathrm{ReLu}(W_1 \cdot x + b_1)$$

The hidden layer $h[n_h]$ can in turn be transformed to other sets of values up to the last transformation that provides the output values y. In this case:

$$y = (W_2 \cdot h + b_2)$$

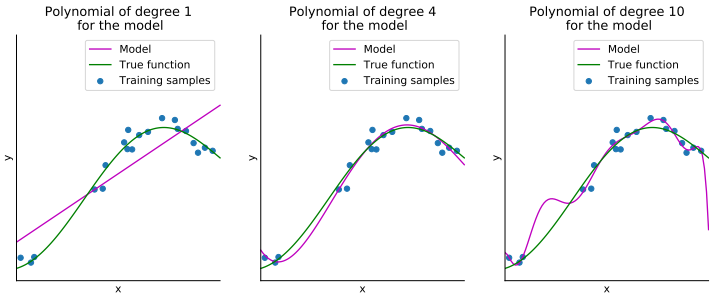


Figure 2.1: Illustration of overfitting and underfitting for a simple 1D regression task in supervised learning (based on one example from the library scikit-learn [Pedregosa et al., 2011]). In the left figure, the degree 1 approximation is underfitting, which means that it is not a good model, even for the training samples; on the right, the degree 10 approximation is a very good model for the training samples but is overly complex and fails to provide a good generalization.

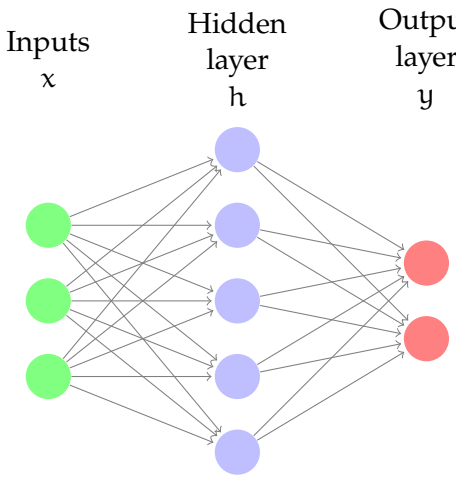


Figure 2.2: Example of a neural network with two fully-connected layers.

with $W_2[n_y \times n_h]$ and $b_2[n_y]$ ($n_y \in \mathbb{N}$).

All these layers are trained in order to minimize a cost function (e.g., RMS error for regression or cross-entropy for classification). The most common methods to learn the levels of abstraction in neural networks are based on gradient descent (the backpropagation algorithm [Rumelhart et al., 1988]), which allows the algorithm to change its internal parameters θ so as to fit the desired function. Another approach that has been found successful in some settings is the use of evolutionary strategies (e.g., in the RL context [Salimans et al., 2017]). A combination of gradient descent and evolutionary strategies can also provide some strong advantages [Fernando et al., 2016; Real et al., 2017; Miikkulainen et al., 2017].

In current applications, many different types of layers have appeared, each providing specific advantages depending on the appli-

cation. We merely cite here two types of layers that are of particular interest:

- Convolutional layers are particularly well suited for images and sequential data (see Fig 2.3 and [LeCun et al., 1995] for a complete description), mainly due to their property of translation invariance (the same learnable filter is applied everywhere on the input feature map). In fact, they are a particular kind of feedforward layer where the difference with the fully-connected layer is that many weights are set to 0 (not learnable) and that other weights share the same value.

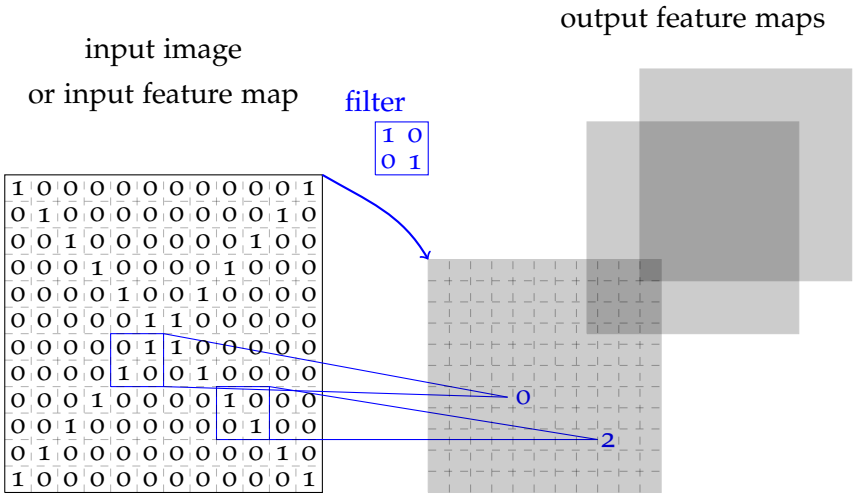


Figure 2.3: Illustration of a convolutional layer with one input feature map that is convolved by different filters to yield the output feature maps. The parameters that are learned for this type of layer are those of the filters. For illustration purposes, some results are displayed for one of the output feature maps with a given filter (in practice, that operation is followed by a non-linear activation function).

- Recurrent layers are particularly well suited on sequential data (see Fig 2.4). Many different variants can be of great interest depending on the context. The "long short-term memory" networks (LSTMs) [Hochreiter and Schmidhuber, 1997] have gained a lot of interest because they are able to work on longer sequences than basic recurrent layers. Over the last few years, other variants with new properties have gained interest. For instance, Neural Turing Machines (NTMs) [Graves et al., 2014] are provided with differentiable "external memory" and are well suited for inferring even longer-term dependencies than LSTMs with low degradation.

Several other specific neural network architectures have also been studied to improve generalization in deep learning. (i) It is possible to design an architecture in such a way that it automatically focuses

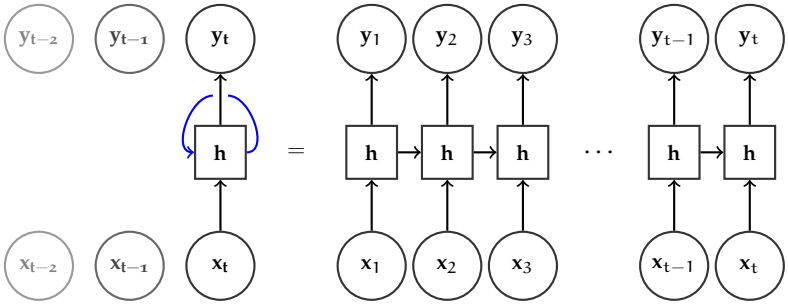


Figure 2.4: Illustration of a simple recurrent neural network. The layer denoted by "h" may represent any non linear function that takes two inputs and provides two outputs. On the left is the simplified view of a recurrent neural network that is applied recursively to (x_t, y_t) for increasing values of t and where the blue line presents a delay of one time step. On the right, the neural network is unfolded with the implicit requirement of presenting all inputs and outputs simultaneously.

on only some parts of the inputs with a mechanism called "attention" (e.g., [Xu et al., 2015; Vaswani et al., 2017]). (ii) Other approaches aim at working with symbolic rules by learning to represent and execute programs [Reed and De Freitas, 2015] and by creating programs where the supervision is in the form of correct inputs/outputs of the task [Neelakantan et al., 2015; Johnson et al., 2017; Chen et al., 2017].

In this introduction, we do not cover in depth the technical part of deep learning, and the interested reader willing to have a review of deep learning techniques can refer to [LeCun et al., 2015; Schmidhuber, 2015; Goodfellow et al., 2016] as well as references therein.

We conclude this section by giving a sense of what deep learning is able to achieve in the fields of supervised learning, unsupervised learning and reinforcement learning.

DEEP LEARNING IN SUPERVISED LEARNING In the field of supervised learning, deep learning has already exceeded human capabilities in most types of tasks where enough data can be provided. As an illustration, Figure 2.5 shows the error rate of both humans and computers in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), which is the most well-known competition in the field of image recognition.

DEEP LEARNING IN UNSUPERVISED LEARNING The field of unsupervised learning has also seen dramatic improvements, thanks to the use of deep learning. One of the most promising approaches in the field is the Generative Adversarial Networks (GAN) architecture [Goodfellow et al., 2014]. The latest results in the field demonstrate that these types of algorithms achieve impressive results on image

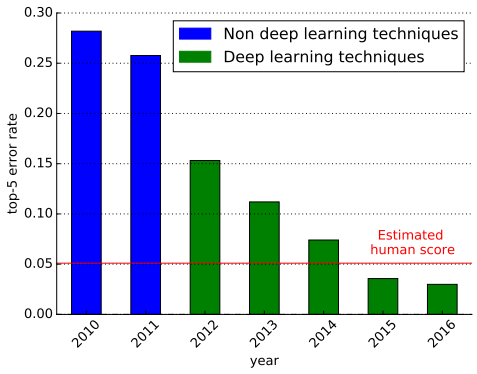


Figure 2.5: Top-5 error rate on the ILSVRC challenge [Russakovsky et al., 2015].

generation tasks, at resolutions up to 256×256 pixels, as can be seen in Figure 2.6.

Figure 2.6: Sample of generated images in an unsupervised way from a dataset of 360K celebrity faces. This picture is taken from [Berthelot et al., 2017].

DEEP LEARNING IN REINFORCEMENT LEARNING Lastly, the combination of RL and deep learning has recently shown its ability to learn how to complete complex tasks from high-dimensional inputs that were previously believed to be extremely difficult for a computer. Among other fruitful applications, this family of algorithms is attaining superhuman-level performance in playing ATARI games from the pixels [Mnih et al., 2015] or mastering the game of Go [Silver et al., 2016a]. It has already shown significant potential for real-world applications, such as robotics [Levine et al., 2016] and smartgrids as discussed in the second part of this thesis.

In the following of this chapter, we will focus on the RL framework and introduce how deep learning and RL can be used together.

2.3 INTRODUCTION TO REINFORCEMENT LEARNING

Reinforcement learning (see [Sutton and Barto, 2017] for an in-depth overview) is, as already noted, the area of machine learning for sequential decision-making under uncertainty. The key aspect of RL is that it uses previous trial-and-error experience to learn a good behavior. In that aspect, it is different from the idea of evolutionary methods (genetic algorithms, genetic programming, simulated annealing, etc.), since, in that case, the key aspect is to rely on many non-learning agents where only the most promising genes (according to some objectives) are transmitted with reproduction, mutation or recombination to the next generation ([Fogel, 2006] for an overview). Note that combining RL and evolutionary methods has also received attention in the literature (e.g., [Whiteson and Stone, 2006; Singh et al., 2010]).

The RL problem is formalized as an agent that has to make decisions in an environment to optimize a given notion of cumulative rewards. As will be discussed, this framework applies to a wide variety of tasks and captures many essential features of artificial intelligence such as a sense of cause and effect as well as a sense of uncertainty and nondeterminism.

2.3.1 *Formal framework*

One of the simplest problems conceptually is the k-armed bandit problem (so named by analogy to a slot machine, except that it has k levers instead of one) where the agent is faced with a choice among k different options, or actions. After each choice, the agent receives a numerical reward chosen from a stationary probability distribution which depends on the action that the agent has selected. This type of sequential decision-making has received a lot of attention in the literature (see [Kuleshov and Precup, 2014] for a review of the most popular multi-armed bandit algorithms) and has many applications in the context of recommendations in clinical trials, marketing, etc.

The k-armed bandit problem is however a limiting case of a larger family of problems and in the general case, actions may affect the next situation (the agent does not stay in front of the same k levers).

The reinforcement learning framework

The full RL problem is formalized as a discrete time stochastic control process where an agent interacts with its environment in the following way: the agent starts, in a given state within its environment $s_0 \in \mathcal{S}$, by gathering an initial observation $\omega_0 \in \Omega$. At each time step t, the agent has to take an action $a_t \in \mathcal{A}$. As illustrated in Figure 2.7, it follows three consequences: (i) the agent obtains a

reward $r_t \in \mathcal{R}$, (ii) the state transitions to $s_{t+1} \in \mathcal{S}$, and (iii) the agent obtains an observation $o_{t+1} \in \Omega$.

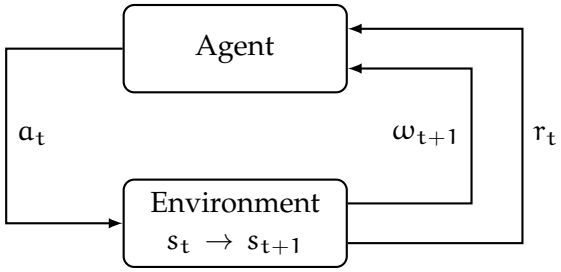


Figure 2.7: Agent-environment interaction in RL.

The Markov setting

For the sake of simplicity, let us in this introduction consider first the case of Makovian stochastic control processes.

Definition 2.1 *A discrete time stochastic control process is Makovian (i.e., it ensures the Markov property) if*

- $\mathbb{P}(o_{t+1} \mid o_t, a_t) = \mathbb{P}(o_{t+1} \mid o_t, a_t, \ldots, , o_0, a_0)$*, and*
- $\mathbb{P}(r_t \mid o_t, a_t) = \mathbb{P}(r_t \mid o_t, a_t, \ldots, , o_0, a_0)$.

The Markov property means that the the future of the process is based solely on the current observation, and the agent has no interest in looking at the full history.

A Markov Decision Process (MDP) is a discrete time stochastic control process defined as follows:

Definition 2.2 *An MDP is a 5-tuple* $(\mathcal{S}, \mathcal{A}, T, R, \gamma)$ *where:*

- $\mathcal{S}$ *is a finite set of states* $\{1, \ldots, N_{\mathcal{S}}\}$,
- $\mathcal{A}$ *is a finite set of actions* $\{1, \ldots, N_{\mathcal{A}}\}$,
- $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ *is the transition function (set of conditional transition probabilities between states),*
- $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathcal{R}$ *is the reward function, where* $\mathcal{R}$ *is a continuous set of possible rewards in a range* $R_{max} \in \mathbb{R}^+$ *(e.g.,* $[0, R_{max}]$*),*
- $\gamma \in [0, 1)$ *is the discount factor.*

The system is fully observable in an MDP, which means that the observation is the same as the state of the environment: $o_t = s_t$. At each time step t, the probability of moving to s_{t+1} is given by the state transition function $T(s_t, a_t, s_{t+1})$ and the reward is given by a bounded reward function $R(s_t, a_t, s_{t+1}) \in \mathcal{R}$. This is illustrated in Figure 2.8.

Note that more general cases than MDPs are introduced in Section 2.5.1 and the partially observable case is formally considered from Chapter 3 onwards.

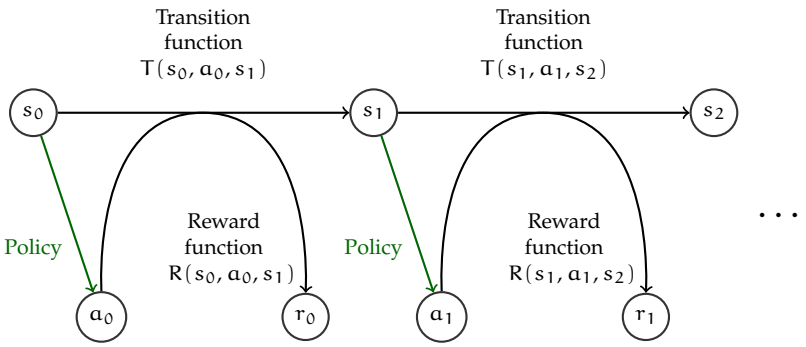


Figure 2.8: Illustration of a MDP.

Different categories of policies

A policy defines how an agent select actions. Policies can be categorized under the criterion of being either stationary or non-stationary. A non-stationary policy depends on the time-step and is mainly useful for finite-horizon RL. In this thesis, the policies considered will be stationary.

Policies can also be categorized under a second criterion of being either deterministic or stochastic:

- In the deterministic case, the policy is described by $\pi(s) : \mathcal{S} \to \mathcal{A}$.
- In the stochastic case, the policy is described by $\pi(s, a) : \mathcal{S} \times \mathcal{A} \to [0, 1]$ where $\pi(s, a)$ denotes the probability that action a may be chosen in state s.

The expected return

In an MDP $(\mathcal{S}, \mathcal{A}, T, R, \gamma)$, let us consider the case of an RL agent whose goal is to find a deterministic policy $\pi(s) \in \Pi$, so as to optimize an expected return $V^\pi(s) : \mathcal{S} \to \mathbb{R}$ (also called V-value function) such that

$$V^\pi(s) = \mathbb{E}\left[\sum_{k=0}^{H-1} \alpha_k r_{t+k} \mid s_t = s, \pi\right], \tag{2.2}$$

where:

- $r_t = R\big(s_t, \pi(s_t), s_{t+1}\big)$,
- $\mathbb{P}\big(s_{t+1}|s_t, \pi(s_t)\big) = T(s_t, \pi(s_t), s_{t+1})$,
- $H \in \mathbb{N}$ is the horizon (infinite horizons will be used in this thesis at the exception of experiments where finite horizons will be used for computational reasons), and
- α_k denotes time-step dependent weighting factors, set to $\alpha_k = \gamma^k$ for discounted RL [1].

1 Note that an alternative is to have $\alpha_k = 1/H$ for the average reward case.

From the definition of the expected return, the optimal expected return can be defined as:

$$V^*(s) = \max_{\pi \in \Pi} V^\pi(s). \tag{2.3}$$

2.3.2 *Different components to learn a policy*

In general, an RL agent may include one or more of the following components:

- a representation of a value function that provides a prediction of how good is each state or each couple state/action,
- a direct representation of the policy $\pi(s)$ or $\pi(s, a)$, or
- a model of the environment in conjunction with a planning algorithm.

The first two components are often related to what is called "model-free" RL and they are discussed in sections 2.3.4, 2.3.5 and 2.3.6. When the latter component is used, the algorithm is referred to as "model-based" RL, which is discussed in Section 2.3.7. A combination of both is discussed in Section 2.3.8. A schema with all possible approaches is provided in Figure 2.9.

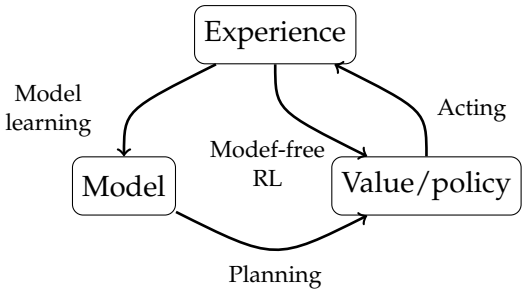


Figure 2.9: General schema of the different methods for RL.

2.3.3 *Different approaches to learn a policy from data*

Before going into the details of the different components that can be used, we can already introduce why deep learning is key in RL. The main motivation is to provide generalization capabilities that practically remove the need for an exponential increase of data when adding extra dimensions to the state or action space (curse of dimensionality).

Let us define the following random variables for any given $(s, a) \in \mathcal{S} \times \mathcal{A}$ as

- $T_{(s,a)}$ with possible outcomes $\mathcal{S}$:

$$T_{(s,a)} \sim T(s, a, \cdot)$$

and,

- $R_{(s,a)}$ with possible outcomes $\mathcal{R}$:

$$R_{(s,a)} \sim \sum_{s' \in \mathcal{S}} T(s, a, s') R(s, a, s').$$

The knowledge of the probability distribution of these two random variables allows to know all the interesting information on the model. Yet, in many types of tasks, the agent has to learn from limited data. This appears in two main cases: (i) in the offline learning case where only limited data on a given environment is available and (ii) in an online learning case where, in parallel to learning, the agent gathers experience in the environment with an exploration/exploitation strategy (this will be discussed in Section 2.4.3). Formally, the finite dataset available to the agent $D_s \sim \mathcal{D}_s$ is defined as a set of four-tuples $(s, a, r, s') \in \mathcal{S} \times \mathcal{A} \times \mathcal{R} \times \mathcal{S}$ gathered by sampling independently and identically (i.i.d.)[2]

- a given number of state-action pairs (s, a) from some fixed distribution with $\mathbb{P}(s, a) > 0, \forall (s, a) \in \mathcal{S} \times \mathcal{A}$,
- a next state $s' \sim T(s, a, \cdot)$, and
- a reward $r = R(s, a, s')$.

The particular case of a dataset D_s where the number of tuples tends to infinity is denoted $D_{s,\infty}$.

In a frequentist-based model, for every tuple (s, a), the relative frequencies of occurrence in the dataset D_s for the transition $(s, a) \rightarrow s'$ (resp. the reward $R(s, a, s')$) in the data is used as a measure of the probability distribution of the transitions $T_{(s,a)}$ (resp. the reward $R_{(s,a)}$). Yet, scarcity on the data implies that the frequentist-based model will only be approximate. Indeed, the frequencies of occurrence in a dataset are themselves random variables. If the limited data have been gathered on the exact same task, it is an unbiased estimator of the true transition probabilities of $(s, a) \rightarrow s'$ (or the reward $R(s, a, s')$), yet a variance is associated to those estimates.

Two different approaches are possible. The Bayesian approach explicitly models the uncertainty on the environment and provides an elegant approach to this problem. It has two key advantages (see [Ghavamzadeh et al., 2015] for an overview): (i) it provides an elegant approach to the exploration/exploitation strategy in the online setting, and (ii) it allows incorporating prior knowledge if available in a principled way. The difficulty is that except for a limited

2 That i.i.d. assumption can, for instance, be obtained from a given distribution of initial states by following a stochastic sampling policy that ensures a non-zero probability of taking any action in any given state. That sampling policy should be followed during at least H time steps with the assumption that all states of the MDP can be reached in a number of steps smaller than H from the given distribution of initial states.

set of bandit environments, it is intractable to compute the Bayesian-optimal strategy and one has to rely on heuristics. The other approach is based on frequentist statistics without modeling explicitly the uncertainty on the environment.

Deep RL is commonly derived from the latter approach. For most problems approaching real-world complexity, the state space are high-dimensional (and possibly continuous). In order to learn the model, the value function or directly the policy, there are two main advantages for RL algorithms to rely on deep learning:

- Neural networks are well suited for dealing with high-dimensional sensory inputs (such as times series, frames, etc.) without requiring an exponential increase of data (see Section 2.2).
- In addition, they work readily online, which means that they can make use of additional samples obtained as learning happens by incrementally improving the targeted function approximator(s).

2.3.4 *Value-based methods*

This class of algorithms aims to build a value function, which subsequently allows to define a policy. Many of these algorithms do not work directly with the V-value function. Instead, they make use of a Q-value function $Q^\pi(s,a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ which is defined as follows:

$$Q^\pi(s,a) = \mathbb{E}\left[\sum_{k=0}^{H-1} \gamma^k r_{t+k} \mid s_t = s, a_t = a, \pi\right]. \tag{2.4}$$

Note that this equation can be rewritten recursively in the case of an MDP:

$$Q^\pi(s,a) = \sum_{s' \in S} T(s,a,s') \left(R(s,a,s') + \gamma Q^\pi(s', a = \pi(s'))\right). \tag{2.5}$$

Similarly to the V-value function, $Q^*(s,a)$ can also be defined as

$$Q^*(s,a) = \max_{\pi \in \Pi} Q^\pi(s,a). \tag{2.6}$$

The particularity of the Q-value function as compared to the V-value function is that the optimal policy may be obtained from $Q^*(s,a)$ in a fully model-free way:

$$\pi^*(s) = \operatorname*{argmax}_{a \in \mathcal{A}} Q^*(s,a). \tag{2.7}$$

An illustration of $V^*(s)$ and $Q^*(s,a)$ is provided in Figure 2.10.

We discuss hereafter one of the simplest and most popular value-based algorithm: the Q-learning algorithm. We also specifically discuss the main elements of DQN [Mnih et al., 2015] (and its

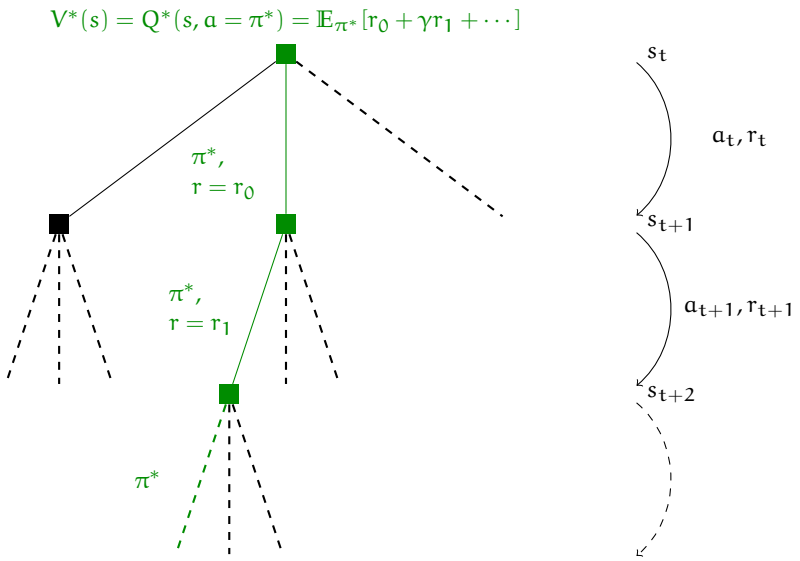


Figure 2.10: Value function illustration.

variants) which is the adaptation of the Q-learning algorithm with neural networks as function approximators. We then discuss how the Q-learning algorithm compares to other value-based methods and provide specific resources for the details.

Q-learning algorithm

The basic version of Q-learning keeps a lookup table of values $Q(s, a)$ with one entry for every state-action pair. In order to build the optimal Q-value function, the Q-learning algorithm makes use of the fixed-point iteration of the Bellman equation for the Q-value function [Bellman and Dreyfus, 1962] which has as a (unique) solution $Q^*(s, a)$:

$$Q^*(s, a) = (\mathcal{B}Q^*)(s, a), \tag{2.8}$$

where $\mathcal{B}$ is the Bellman operator mapping any function $K : S \times A \to \mathbb{R}$ into another function $S \times A \to \mathbb{R}$ and is defined as follows for an MDP:

$$(\mathcal{B}K)(s, a) = \sum_{s' \in S} T(s, a, s') \left(R(s, a, s') + \gamma \max_{a' \in A} K(s', a') \right). \tag{2.9}$$

The fixed point of Equation 2.8 exists since the Bellman operator is a contraction mapping[3]. In practice, when the environment is only known from experience, a general proof of convergence to the

3 The Bellman operator is a contraction mapping because it can be shown that for any pair of bounded functions $K, K' : S \times A \to \mathbb{R}$, the following condition is respected:

$$\|TK - TK'\|_\infty \leqslant \gamma \|K - K'\|_\infty.$$

optimum is available [Watkins and Dayan, 1992] at the conditions that:

- the state-action pairs are represented discretely, and
- all actions are repeatedly sampled in all states.

As discussed earlier, this simple setting is often inapplicable due to the high-dimensional (possibly continuous) state-action space. In that context, a parameterized value function $Q(s, a; \theta_k)$ is needed, where θ_k refers to the parameters at the k^{th} iteration.

Experiences are gathered in the form of tuples (s, a, r, s') where the reward r and the state at the next time-step s' follows the state-action pair (s, a). At every iteration, the current value $Q(s, a; \theta_k)$ is updated towards a target value

$$Y_k^Q = r + \gamma \max_{a' \in A} Q(s', a'; \theta_k). \tag{2.10}$$

The Q-learning update when using the squared-loss amounts in updating the parameters:

$$\theta_{k+1} = \theta_k + \alpha(Y_k^Q - Q(s, a; \theta_k))\nabla_{\theta_k} Q(s, a; \theta_k), \tag{2.11}$$

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

The Q-learning rule from Equation 2.11 can be directly implemented online using a neural network $Q(s, a; \theta_k)$ to aim at convergence towards $Q^*(s, a)$, where the parameters θ_k may be updated by stochastic gradient descent (or a variant). However, due to the generalization and extrapolation abilities of neural networks, they can build unpredictable changes at different places in the state-action space. Therefore, the contraction mapping property of the Bellman operator in Equation 2.9 is not enough to guarantee convergence. It is verified experimentally that these errors may propagate with this online update rule and, as a consequence, convergence may be slow or even unstable [Baird, 1995; Tsitsiklis and Van Roy, 1997; Gordon, 1999; Riedmiller, 2005] [4]. Another related damaging side-effect of using function approximators is the fact that Q-values tend to be overestimated due to the argmax operator [Hasselt, 2010]. Because of the instabilities and the risk of overestimation, specific care has be taken to ensure proper learning.

The deep Q-learning algorithm introduced in [Mnih et al., 2015] uses two important elements: a target Q-network and a replay memory.

- The target Q-network in Equation 2.10 is replaced by $Q(s', a'; \theta_k^-)$ where its parameters θ_k^- are updated only every C iterations with the following assignment: $\theta_k^- = \theta_k$.

4 Note that this drawback does not happen when using kernel-based regressors (such as k-nearest neighbors, linear and multilinear interpolation, etc.) [Gordon, 1999] or tree-based ensemble methods [Ernst et al., 2005]. However, these methods have not proved able to handle successfully high-dimensional inputs.

- The replay memory keeps all information for the last N_{replay} time steps. The updates are then made on mini-batches of tuples (s, a, r, s') selected within the replay memory.

Those elements limit instabilities or risks of divergence when a nonlinear function approximator such as a neural network is used in combination with value-based methods. A sketch of the algorithm is given in Figure 2.11.

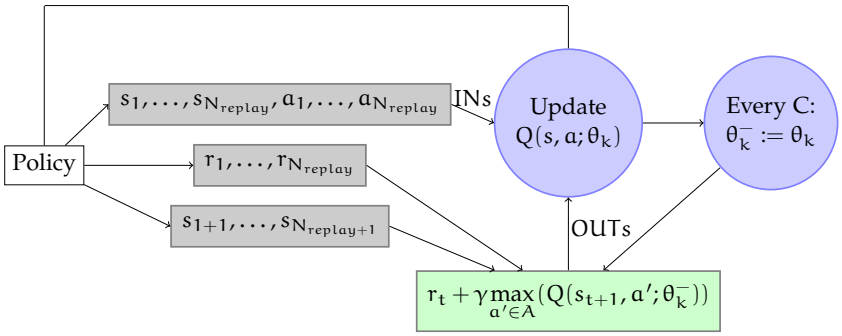


Figure 2.11: Sketch of the DQN algorithm. $Q(s, a; \theta_k)$ is initialized to close to 0 everywhere on its domain; N_{replay} is the size of the replay memory; the target Q-network parameters θ_k^- are only updated every C iterations with the Q-network parameters θ_k and are held fixed between updates; the variable INs corresponds to a mini-batch (e.g., 32 elements) of tuples (s, a) taken randomly in the replay memory and the variable OUTs is the corresponding mini-batch of target values for the tuples.

Many additional improvements can be found in the literature and a few of them are cited hereafter. In [Wang et al., 2015], the neural network architecture decouples value and advantage function (defined as $A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$) which leads to improved performance. In [Van Hasselt et al., 2015], a specific update scheme (variant to Equation 2.11) leads to less overestimation of the Q-learning values, as well as improved stability, hence improved performance. Parallel actor-learners [Mnih et al., 2016] or the use of unsupervised auxiliary tasks [Jaderberg et al., 2016] also lead to faster and more robust learning. In [Pritzel et al., 2017], a differentiable memory module allows to rapidly integrate recent experience by interpolating between Monte Carlo value estimates and backed up off-policy estimates. Additional elements to improve learning are discussed in Chapter 4.

How does Q-learning compare to other value-based methods

The Q-learning algorithm is characterized by the following two characteristics:

- Q-learning is off-policy, which means that it builds a value function (or in general a policy) independently of the policy that is used to gather experiences. On the contrary, on-policy

methods attempt to evaluate or improve the policy that is used to make decisions (e.g., SARSA algorithm [Rummery and Niranjan, 1994]).

- Q-learning uses bootstrapping, which means that it updates the value estimates of future return from raw experiences by using its own value estimate recursively (without waiting for final outcome). Non-bootstrapping methods learn directly from returns (Monte Carlo). An intermediate solution is the use of eligibility traces [Singh and Sutton, 1996]. On the one hand, value bootstrap methods are more readily applied to off-policy data which is appealing because it can learn while exploring safely. On the other hand, non-bootstrapping methods are better behaved when combined with function approximation (no instability) and propagate information more quickly from delayed rewards. The interested reader can refer to [Munos et al., 2016] for a discussion on the conditions required to learn efficiently and safely from returns in the context of off-policy learning.

2.3.5 *Policy-based methods*

The basic idea behind these algorithms is to learn a parameterized policy π_θ. When the policy is parametrized with a neural network, the usual approach is to adjust the parameters θ in a direction that aims to increase the expected return: $\Delta\theta \propto \nabla_\theta V^{\pi_\theta}(\cdot)$. These policy gradient algorithms have the following interesting properties as compared to value-based methods:

- they are able to work with continuous action space, and
- they can learn stochastic policies.

The main equations related to stochastic policy gradient and deterministic policy gradient are introduced hereafter.

2.3.5.1 *Stochastic policy gradient*

The expected return of a stochastic policy π starting from state s_0 from Equation 2.2 can be written as:

$$V^\pi(s_0) = \int_{\mathcal{S}} \rho^\pi(s) \int_{\mathcal{A}} \pi(s,a) R'(s,a) \mathrm{d}a\mathrm{d}s, \tag{2.12}$$

where $R'(s,a) = \sum_{s' \in \mathcal{S}} T(s,a,s') R(s,a,s')$ and $\rho^\pi(s)$ is the discounted state distribution defined as

$$\rho^\pi(s) = \sum_{t=0}^{\infty} \gamma^t \Pr\{s_t = s | s_0, \pi\}.$$

For a differentiable policy π_θ, the fundamental result underlying these algorithms is the policy gradient theorem (Sutton et al., 1999):

$$\nabla_\theta V^{\pi_\theta}(s_0) = \int_{\mathcal{S}} \rho^{\pi_\theta}(s) \int_{\mathcal{A}} \nabla_\theta \pi_\theta(s,a) Q^{\pi_\theta}(s,a) da ds. \tag{2.13}$$

Note that this result is particularly interesting since the policy gradient does not depend on the gradient of the state distribution (even though one could have expected it to).

The following identity known as the likelihood ratio trick can then be exploited:

$$\begin{aligned} \nabla_\theta \pi_\theta(s,a) &= \pi_\theta(s,a) \frac{\nabla_\theta \pi_\theta(s,a)}{\pi_\theta(s,a)} \\ &= \pi_\theta(s,a) \nabla_\theta \log(\pi_\theta(s,a)). \end{aligned} \tag{2.14}$$

Thanks to Equation 2.14 and assuming that trajectories are generated from a system by roll-outs following policy π_θ, it follows that

$$\nabla_\theta V^{\pi_\theta}(s_0) = \mathbb{E}_{s\sim\rho^\pi, a\sim\pi_\theta} \left[\nabla_\theta \left(\log \ \pi_\theta \right) Q^{\pi_\theta}(s,a) \right]. \tag{2.15}$$

Note that it is common to add an entropy regularizer to the gradient in order to prevent the policy from becoming deterministic. It is also interesting to replace the Q-value function by the advantage function $A^{\pi_\theta}(s,a) = Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s,a)$ because this allows to have a lower variance in the gradient estimate [Schulman et al., 2015b; Gu et al., 2016]. An interesting idea is the one provided with the algorithm TRPO that provides policy gradients with a KL divergence constraint to prevent the new policy from diverging too far from the existing policy [Schulman et al., 2015a].

One of the remaining challenges for policy gradient using function approximator is how to estimate the action-value function $Q^{\pi_\theta}(s,a)$ or the advantage function $A^{\pi_\theta}(s,a)$. One straightforward possibility is to estimate it with the return of rollouts on the environment while following policy π_θ ("Monte Carlo policy gradient" or REINFORCE algorithm [Williams, 1992]). Even though they are well-behaved when used in conjunction with back-propagation of a neural approximator of the policy, the main drawback is the large variance of the estimator. Moreover they require a priori on-policy rollouts. Another possibility is to use an actor-critic method as will be discussed in Section 2.3.6.

2.3.5.2 *Deterministic policy gradient*

The policy gradient methods may also be extended to deterministic policies. Let us denote by $\pi(s)$ the deterministic policy : $\pi(s) : \mathcal{S} \rightarrow \mathcal{A}$.

In discrete action spaces, a direct approach is to build the policy iteratively with:

$$\pi_{k+1}(s) = \underset{a\in\mathcal{A}}{\operatorname{argmax}} \ Q^{\pi_k}(s,a), \tag{2.16}$$

where π_k is the policy at the k^{th} iteration.

In continuous action spaces, a greedy policy improvement becomes problematic, requiring a global maximisation at every step. Instead, let us denote by $\pi_\theta(s)$ a differentiable deterministic policy. In that case, a simple and computationally attractive alternative is to move the policy in the direction of the gradient of Q [Silver et al., 2014] which leads to the Deep Deterministic Policy Gradient (DDPG) algorithm [Lillicrap et al., 2015] :

$$\nabla_\theta V^{\pi_\theta}(s_0) = \mathbb{E}_{s\sim\rho^{\pi_\theta}}\left[\nabla_\theta(\pi_\theta)\nabla_a(Q^{\pi_\theta}(s,a))|_{a=\pi_\theta(s)}\right]. \quad (2.17)$$

This implies relying on $\nabla_a(Q^{\pi_\theta}(s,a))$ (in addition to $\nabla_\theta\pi_\theta$) which usually requires using actor-critic methods (see Section 2.3.6).

2.3.6 *Actor-Critic methods*

Vanilla actor-critic consists of two eponymous components: (i) an actor that adjusts the parameters θ of the policy $\pi_\theta(s)$ while (ii) a critic adjusts the parameters w of an action-value function $Q^{\pi_\theta}(s,a;w)$. Current state of the art results using an actor-critic architecture are described in [Gruslys et al., 2017].

Note that it has recently been shown that optimizing the policy or the value function can be shown to be intimately related in some specific settings, depending on the loss function and on the entropy regularization used [O'Donoghue et al., 2016; Schulman et al., 2017]. In fact, all model-free methods can thus be thought of as different facets of the same concept.

2.3.7 *Model-based methods*

In model-based approaches, the model is either explicitly given (e.g., in the game of GO for which all the rules are known a priori) or it can be learned from experience. To learn the model, the usage of function approximators brings yet again significant advantages with high-dimensional (possibly partially observable) environments such as in [Oh et al., 2015; Mathieu et al., 2015; Finn et al., 2016a; Kalchbrenner et al., 2016; Duchesne et al., 2017]. The model can then act as a proxy for the actual environment.

When a generative model of the environment is available, planning consists in interacting with the model using lookahead search. Monte-Carlo tree search (MCTS) techniques [Browne et al., 2012] are among the most popular approaches. They have gained popularity, among other things, thanks to prolific achievements in the challenging task of computer Go [Brügmann, 1993; Gelly et al., 2006; Silver et al., 2016a]. The idea is to sample multiple trajectories from the current state until a terminal condition is reached (e.g., a given maximum

depth), and then to recommend an action based on those samples. The main difficulty in sampling trajectories is to balance exploration and exploitation. On the one hand, the purpose of exploration is to gather more information on the part of the search tree where few simulations have been performed (i.e., where the expected value has a high variance). On the other hand, the purpose of exploitation is to refine the expected value of the currently most promising moves. One well-known simple but effective method for balancing exploitation and exploration is called UCT [Kocsis and Szepesvári, 2006] (which is based on UCB [Auer et al., 2002]).

2.3.8 *Integrating learning and planning (i.e., model-free and model-based methods)*

Usually, RL algorithms have been categorized as being either model-based or model-free. The respective strengths of the two approaches depend on different factors.

First, it depends on the way the agent has access to the environment: (i) it can either directly have access to a generative model of the environment (in the form of a simulator with the possibility to gather data infinitely and in a flexible way), or (ii) it can know the environment only through a (possibly finite) number of trajectories within that environment. In the latter case, a model-based approach is possible but requires an additional step of model estimation. Note that learning the model can share the hidden-state representations with a value-based approach [Li et al., 2015].

Second, the model-based approach requires a priori working in conjunction with a planning algorithm, which is often computationally demanding. When the model is available to the agent, a pure planning approach can theoretically be optimal but would often be impractical due to time constraints when computing the policy $\pi(s)$ (e.g., for applications with real-time decision-making or simply due to resource limitations).

Third, for some tasks, the structure of the policy (or value function) is the easiest one to learn but for other tasks, the model of the environment may be learned more efficiently due to the particular structure of the task (less complex/more regularity). Thus, the most adapted approach also depends on the structure of the model and on the structure of the policy/value function. Let us consider two examples to better understand this key consideration. In a labyrinth where the agent has full observability, it is clear how actions affect the next state and the dynamic of the model may easily be generalized by the agent from few tuples (i.e., the agent is blocked when trying to cross a wall of the labyrinth, or it goes forward when trying to go in an unblocked direction). Once the model is known, a planning algorithm can then be used with high performance. Let us now discuss another example where, on the contrary, planning is more

difficult: an agent has to cross a road with random events happening everywhere on the road and let us suppose that the best policy is simply to move forward except when an object has just appeared in front of the agent. In that case, the simple policy of moving forward except for specific observations may be easily captured by a model-free approach while developing a model of the environment and plan within it would be more difficult (mainly due to the stochasticity of the model which leads to many different possible situations, even for one given sequence of actions).

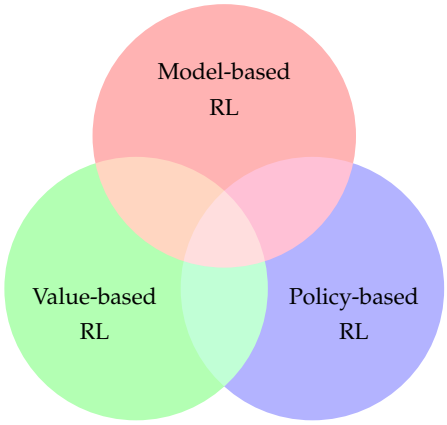


Figure 2.12: Venn diagram of the different types of RL algorithms.

As illustrated in Figure 2.12, there is a possibility to obtain advantages from both worlds by integrating learning and planning into one end-to-end training procedure so as to obtain an efficient algorithm both in performance (sample efficient) and in computation time.

When the model is available, one direct approach is to use tree search techniques that make use of value and policy networks (e.g., [Silver et al., 2016a]). If the model is differentiable, one can also directly compute an analytic policy gradient by backpropagation of rewards along trajectories [Nguyen and Widrow, 1990].

When the model is not available and in the assumption that the agent has only access to a limited number of trajectories, the key property is to have an algorithm that generalizes well (see Section 2.4.1 for a discussion on generalization). Methods such as the Dyna-Q method [Sutton, 1990] uses a learned model to generate additional samples for the model-free algorithm (see also [Gu et al., 2016] for an extension in deep RL). A recent work called the predictron [Silver et al., 2016b] aims at developing a learnable internal model that is effective in the context of planning. In the same spirit, in [Tamar et al., 2016], a fully differentiable neural network with a planning module embedded within is introduced which provides an elegant approach to improve generalization with the integration of learning and planning. Other works have also build architectures that combine model-based and model-free [Oh et al., 2017; Weber et al., 2017]. Yet

other works have developed a strategy to directly learn end-to-end the model along with how to make the best use of it without relying on explicit tree search techniques [Pascanu et al., 2017].

2.4 CHALLENGES IN DEEP REINFORCEMENT LEARNING

2.4.1 *Generalization and transfer learning*

Generalization is a central concept in the whole field of machine learning and reinforcement learning is no exception. In an RL algorithm (model-free or model-based), generalization refers to either

- the capacity to obtain a good performance in an environment where previous limited data has been gathered, or
- the capacity to obtain a good performance in a slightly different environment.

The former case is discussed in Section 2.4.1.1 and the latter is discussed in Section 2.4.1.2.

2.4.1.1 *Generalization with limited data*

In the case where the agent has to learn how to behave in the same environment it has been trained on, the idea of generalization is direclty related to sample efficiency and it refers to the capacity to learn a task without requiring a huge amount of experience in the environment. Let us consider the case of a limited dataset D_s obtained on the exact same task. The process of building a policy can be seen as a function mapping a dataset D_s into a policy π_{D_s} (independently of whether the policy comes from a model-based or a model-free approach). The suboptimality of the expected return can be decomposed as follows:

$$\begin{aligned}\mathop{\mathbb{E}}_{D_s \sim \mathcal{D}_s}[V^{\pi^*}(s) - V^{\pi_{D_s}}(s)] = &\underbrace{(V^{\pi^*}(s) - V^{\pi_{D_{s,\infty}}}(s))}_{\text{asymptotic bias}} \\ &+ \underbrace{\mathop{\mathbb{E}}_{D_s \sim \mathcal{D}_s}[(V^{\pi_{D_{s,\infty}}}(s) - V^{\pi_{D_s}}(s))\ ]}_{\substack{\text{error due to finite size of the dataset } D_s \\ \text{referred to as overfitting}}}. \end{aligned} \tag{2.18}$$

This decomposition highlights two different terms: (i) an asymptotical bias which is independent of the quantity of data and (ii) an overfitting term directly related to the fact that the amount of data is limited. The goal of building a policy π_{D_s} from a dataset D_s is to obtain the lowest overall suboptimality which, as we will see, will require to find a good tradeoff between these two terms.

A technical in-depth analysis of this question is provided in Chapter 3 in the context of the partially observable setting and limited

data but a simple example showing how deep RL deals with the frequentist assumption is provided hereafter. This simple example is, by no means, representative of the complexity of the real-world problems but it is enlightening to simply illustrate the concepts that will be discussed more formally in the following.

Let us consider an MDP with $N_{\mathcal{S}} = 9 + 2$ and $N_{\mathcal{A}} = 4$ illustrated in Figure 2.13. Let us suppose that the main part of the environment is a discretized square domain represented by 3×3 states (each represented by a tuple (x,y) with x={0,1,2}, y={0,1,2}). The agent starts in the central state $(1, 1)$. In every state, it selects one of the 4 actions corresponding to the 4 directions (up, down, left and right), which leads the agent to transition deterministically in that direction to the state immediately next to it, except when it tries to move out of the domain. On the upper part and lower part of the domain, the agent is stuck in the same state if it tries to move out of the domain. On the left, the agent transitions deterministically to a given state that will provide, at the next time step, a reward of 0.5 for any action while on the right, the agent transitions with a probability 25% to another state that will provide, at the next time step, a reward of 1 for any action (the rewards are 0 for all other states). When a reward is obtained, the agent transitions back to the central state.

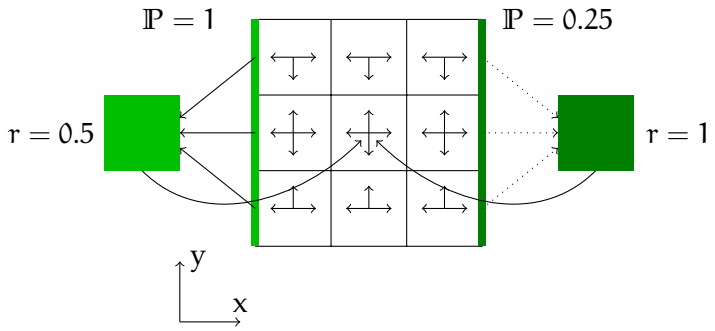


Figure 2.13: Representation of a simple MDP that illustrates the need of generalization.

In this example, it is obvious (if the agent has perfect knowledge of its environment) that the best expected cumulative reward (for a discount factor close to 1) would be to always go to the left direction and repeatedly gather the 0.5 reward. Let us now suppose that only limited information has been obtained on the MDP with only one tuple of experience (s, a, r, s') for each couple (s, a). According to the limited data, there is a rather high probability ($\sim$ 58%) that at least one transition from the right side seems to provide a deterministic access to $r = 1$. An agent that would use the frequentist approach of the training data without good generalization would thus have a suboptimal policy in these cases.

Here are three key elements that are at stake when one wants to improve generalization:

1. Feature selection: When using non-informative features on which to base the policy (in the example the y-coordinate of the state), an RL algorithm may take into consideration spurious correlations. In other words, it may base its decisions on events or variables that are not dependent on the actual goal that the agent seeks. Yet, it may be wrongly inferred that they are correlated and lead to overfitting (in the example, the agent may infer that the y-coordinate changes something to the expected return). On the contrary, removing important features will introduce an asymptotic bias.

2. Function approximator selection: This abstract level reasoning can also be learned thanks to function approximators. If a too simple function approximator for the value function and/or the policy and/or the model is used, an asymptotic bias may appear because the policy cannot discriminate efficiently the different inputs. While on the other hand, when the function approximator has a poor generalization, there will be a large error due to the finite size of the dataset (overfitting). In the example, a particularly good choice of model-based or model-free approach associated with a good choice of a function approximator could infer that the y-coordinate of the state is less important than the x-coordinate and generalize that to the policy. Note that the function approximator and the selection of the features are linked since the function approximator characterizes how the features will be treated into higher levels of abstraction (a fortiori it can thus give more or less weight to some features). One approach to avoid non-informative features to affect negatively the performance is to force the agent to acquire a set of symbolic rules adapted to the task and reason on a more abstract level (e.g., [Garnelo et al., 2016]). This abstract level reasoning and the improved generalization that it allows are key to implement high-level cognitive functions such as transfer learning, analogical reasoning, etc. For instance, the function approximator may embed a relational learning structure such as in [Santoro et al., 2017] and thus build on the idea of relational reinforcement learning [Džeroski et al., 2001].

3. Optimization horizon: The optimization horizon controls the complexity of the policy class [Petrik and Scherrer, 2009; Jiang et al., 2015b]. There are two consequences. On the one hand, artificially reducing the optimization horizon leads to a potential bias since some (potentially optimal) policies are discarded. On the other hand, if a long optimization horizon is targeted (the discount factor γ is close to 1), there is a higher risk of overfitting. This overfitting can intuitively be understood as linked to the accumulation of the errors in the transitions and rewards estimated from data as compared to the actual transition and reward probabilities. In the example, in the case where the upper

right or lower right states would seem to lead deterministically to $r = 1$ from the limited data, one may take into account that it requires more steps and thus more uncertainty on the transitions (and rewards). In that context, a low training discount factor would lead to a policy that recommends the left action.

As discussed for these three elements, there is in practice a tradeoff to be made between asymptotic bias and overfitting that we simply call "bias-overfitting tradeoff". This is illustrated in Figure 2.14 and will be discussed formally in Chapter 3.

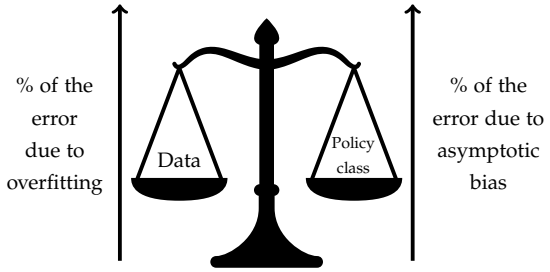


Figure 2.14: Schematic representation of the bias-overfitting tradeoff.

2.4.1.2 *Transfer learning*

Generalization can also be seen as a form of transfer learning and it then refers to the capacity of using efficiently previous knowledge from a source environment to achieve new (slightly) different tasks in a target environment. One of the main use case of transfer learning is to learn a policy in a simulation environment and then use it in a real-world context.

In general, transfer learning can make use of the deep learning architecture to effectively transfer knowledge among tasks by sharing parameters. The direct approach is to pre-train a model/policy/value function in the source environment and then fine-tune it in the target environment [Rusu et al., 2016]. In the same spirit, developments such as [Kirkpatrick et al., 2016] opens up the perspective of learning sequentially different types of tasks in different source environments (while avoiding catastrophic forgetting) before using the policy in a target environment.

Another approach is to use an idea similar to data augmentation in supervised learning so as to make sense of variations that were not encountered in the training data. The idea is that with enough data augmentation on the simulator data, the actual (unseen) task may appear to the agent as just another variation. In [Parisotto et al., 2015], the agent is trained with deep RL techniques on different tasks simultaneously, and it is shown that it can generalize to new related domains. In [Tobin et al., 2017], the agent is trained in a simulated environment while being provided with different renderings and

it is shown that the policy learned transfers well to real images afterwards.

2.4.2 *Hierarchical learning*

The possibility of learning temporally extended actions (as opposed to primitive actions that last for one time-step) has been formalized under the name of options [Sutton et al., 1999] (Similar ideas have also been denoted in the literature as macro-actions or abstract actions). The usage of options is an important challenge in RL because it is essential when the task at hand requires working on long time scales while developing generalization capabilities and easier transfer learning between the "strategies". A few recent works have brought interesting results in the context of fully differentiable (hence learnable in the context of deep RL) options discovery. In [Bacon et al., 2016], an "option-critic" architecture is presented with the capability of learning simultaneously the internal policies and the termination conditions of options, as well as the policy over options. In [Vezhnevets et al., 2016], the deep recurrent neural network is made up of two main elements. The first module generates an action-plan (stochastic plan of future actions) while the second module maintains a commitment-plan which determines when the action-plan has to be updated or terminated.

2.4.3 *Exploration/Exploitation dilemma*

The exploration-exploitation dilemma is a famous tradeoff in RL. As an agent starts accumulating knowledge about its environment, it has to make a tradeoff between learning more about its environment (exploration) or pursuing what seems to be the most promising strategy with the experience gathered so far (exploitation). There exist mainly two different types of cases:

- the agent is expected to perform well without a separate training phase and in that case, an explicit tradeoff between exploration versus exploitation appears so that the agent should explore only when the learning opportunities are valuable enough for the future as compared to what direct exploitation can provide.
- the agent follows a *training policy* during a first phase of interactions with the environment so as to accumulate training data and hence learn a *test policy*. The test policy should then be able to maximize a cumulative sum of rewards in a separate phase of interactions. The goal of the training policy is then to ensure efficient exploration of the state space without constraint directly related to the cumulative reward objective. Note that

> an implicit exploration/exploitation is still important. On the one hand, we have to ensure that the lesser-known parts of the environment are not promising (exploration). On the other hand, the more promising is a part of the environment, the more we are interested in gathering experience in that part of the environment (exploitation) to refine the knowledge of the dynamics.

The exploration techniques are split into two main categories: directed exploration and undirected exploration [Thrun, 1992]. The directed exploration techniques use a memory of the past interactions with the system for guiding the exploration search while the undirected exploration does not use any memory of the past interactions (e.g., ϵ-greedy or softmax exploration which is also called Boltzmann exploration). For finite deterministic domains, directed exploration appears to scale polynomially with the size of the state space while undirected exploration scales in general exponentially with the size of the state space (e.g., E^3 [Kearns and Singh, 2002], R-max [Brafman and Tennenholtz, 2003], ...). One of the key elements in directed exploration is to maximize Shannon information gain thanks to exploration (e.g., [Sun et al., 2011]). Directed exploration is (or at least was) however not easily applicable in the model-free approach with high-dimensional state space (e.g., [Kakade et al., 2003]).

Recently with the development of RL techniques for large-scale state space, some possibilities have been investigated. In [Stadie et al., 2015] or [Houthooft et al., 2016], a directed exploration is encouraged by giving the agent exploration bonuses thanks to heuristics that assess novelty. A similar idea is also introduced in [Bellemare et al., 2016] and it shows promising results on one of the most difficult Atari 2600 game, Montezuma's revenge.

A different line of work in [Kaplan et al., 2017] suggests using natural language to guide the agent by providing exploration bonuses when an instruction is correctly executed. This approach requires however a heavy manual work to provide the dataset (at least instructions and exploration bonuses when correctly executed).

Yet a different line of work in [Florensa et al., 2017] suggests learning useful skills in pre-training environments, which can then be utilized in the actual environment to improve exploration and train a high-level policy over these skills. A method called "Bootstrapped DQN", close to an undirected scheme, has also been proposed. It carries out exploration through the use of randomized value functions which allows a "temporally-extended" exploration [Osband et al., 2016].

A key challenge for the future developments of deep RL algorithms is to handle, for high-dimensional spaces, the exploration/exploitation tradeoff in a principled way.

2.4.4 *Managing experience replay*

A replay memory [Lin, 1992] allows achieving a good data-efficiency by storing the past experience of the agent in order to have the opportunity to reprocess it at a later time. While a replay memory allows processing the transitions in a different order than they are experienced, there is also the possibility to use prioritized replay. This allows considering the transitions with a different frequency than they are experienced depending on their significance (that could be which experiences to store and which ones to replay). In [Schaul et al., 2015], the criterion for prioritized experience replay is based on the magnitude of the transitions' TD error, which indicates how "unexpected" the transitions are.

2.5 DIFFERENT SETTINGS IN REINFORCEMENT LEARNING

2.5.1 *POMDP and learning to learn/meta-learning*

We have so far discussed how an agent is able to learn how to behave in a given Markov environment where all the interesting information (the state $s_t \in \mathcal{S}$) is obtained at every time step t. By definition of the Markov hypothesis, it was straightforward that the policy had no interest to depend on what happened at previous time steps to recommend an action (excluding through the experience gathered in the dataset).

We now describe two different cases that complicate a bit the simple Markov scenario:

- The partially observable scenario: in this setting, the agent only gathers, at each time step, a partial observation of its environment that does not allow to recover the state with certainty.
- The distribution of possible (related) MDPs: in this setting, the agent faces a distribution of different environments that differ for instance in the reward function or on the probabilities of transitions from one state to the other.

Those two problems are at first sight quite different conceptually. However, in both cases, at each step in the sequential decision process, the agent may have a benefit to take into account its whole observable history up to the current time step t when deciding what action to perform. In other words, an history of observations can be used as a pseudo-state (pseudo-state because that refers to a different and abstract stochastic control process). Any missing information in the history of observations (potentially way before time t) can introduce a bias in the RL algorithm.

The POMDP setting will be discussed in depth in Chapter 3 and it is thus not discussed further here. Concerning the distribution

of possible (related) MDPs, the problem that naturally arises is the exploration-exploitation tradeoff. More specifically, the agent can tend to give priority to either gathering valuable information on the domain (exploration) or selecting an action that gives a good expected reward according to the current knowledge of the environment (exploitation). Two different approaches have been investigated in the literature:

- The Bayesian approach has the possibility to explicitly model the distribution of the different MDPs (if a prior is available). As already noted, it is often intractable to compute the Bayesian-optimal strategy and one has to rely on heuristics.
- The concept of "meta-learning" or "learning to learn" aims at discovering, from experience, how to behave in a range of tasks and it is able to negotiate the exploration-exploitation tradeoff [Hochreiter et al., 2001]. In that case, deep RL techniques have been investigated in, e.g., [Wang et al., 2016; Duan et al., 2016] with the approach of using recurrent networks trained on a set of environments drawn i.i.d. from the distribution. This approach has been shown to provide a policy that performs well on problems drawn from that distribution and to a certain extent generalizes to related distributions (see Section 2.4.1 for a discussion on generalization).

2.5.2 *Inverse reinforcement learning and imitation learning*

In some circumstances, the agent is only provided with trajectories, without reward signal, of an expert agent (also called the teacher). Given observed optimal behavior, the question is to know whether the agent can perform similarly. Two approaches are possible:

- Imitation learning uses supervised learning to map the states to the actions from the observations of optimal behavior (e.g., [Giusti et al., 2016]).
- Inverse reinforcement learning (IRL) is the problem of determining a possible reward function given observations of optimal behavior. When the system dynamics is known (except the reward function), this is an appealing approach particularly when the reward function provides the most generalizable definition of the task [Ng et al., 2000; Abbeel and Ng, 2004]. For example, let us consider a large MDP for which the expert always ends up transitioning to the same state. In that context, one may be able to easily infer, from only a few trajectories, what is the probable goal of the task (=a reward function that explains the behavior of the teacher) while directly learning the policy via imitation learning would be much less efficient. Note that recent works bypass the requirement of the knowledge of

the system dynamics [Boularias et al., 2011; Kalakrishnan et al., 2013; Finn et al., 2016b].

A combination of the two approaches has also been investigated in [Neu and Szepesvári, 2012; Ho and Ermon, 2016]. Note also that in real-world applications, the teacher is not exactly in the same context than the agent and transfer learning may be of crucial importance [Schulman et al., 2016; Liu et al., 2017] (see Section 2.4.1.2 for more information on transfer learning).

2.5.3 *Multi-agent systems*

A multi-agent system is composed of multiple interacting agents within an environment. For this type of system, many different settings can be considered.

- Decentralized versus centralized setting: in a decentralized setting, each agent selects its own action conditioned only on its local information. This setting is related to the emergence of communication between agents in order to share information.
- Collaborative versus non-collaborative setting: in a collaborative setting, agents may have a shared reward measurement.

These topics have already been widely investigated and the interested reader can refer to [Foerster et al., 2017; Sunehag et al., 2017] as well as to the related work discussed in these papers.

2.6 APPLYING REINFORCEMENT LEARNING TO REAL-WORLD PROBLEMS

2.6.1 *Successes of reinforcement learning*

Deep RL techniques have demonstrated their ability to tackle a wide range of problems that were previously unsolved. Some of the most renowned achievements are

- attaining superhuman-level performance in playing ATARI games from the pixels [Mnih et al., 2015],
- mastering the game of Go [Silver et al., 2016a],
- beating professional poker players in the game of heads up no-limit Texas hold'em: Libratus [Brown and Sandholm, 2017] and Deepstack [Moravčík et al., 2017], as well as
- beating world's top professionals at 1v1 matches of Dota 2 (real-time strategy game).

These achievements in popular games are important mainly because they show the potential of deep RL in a wide variety of complex and diverse tasks that require working from high-dimensional inputs. In fact, deep RL has also already shown lots of potential for real-world applications such as robotics [Levine et al., 2016; Gandhi et al., 2017], self-driving cars [You et al., 2017], locomotion skills [Peng et al., 2017], online marketing [Pednault et al., 2002], finance [Deng et al., 2017], smartgrids (this thesis), etc. Virtual environments such as the ones developed by OpenAI [Brockman et al., 2016] and DeepMind [Beattie et al., 2016] will certainly allow to explore even further the limits of the deep RL algorithms.

RL has also applications in fields where one could think that supervised learning alone is sufficient such as sequence prediction (e.g., text) [Ranzato et al., 2015; Bahdanau et al., 2016]. Designing the right neural architecture for supervised learning tasks has also been cast as an RL problem (e.g., [Zoph and Le, 2016] that uses REINFORCE). Note that those types of tasks can also be tackled with evolutionary strategies [Miikkulainen et al., 2017; Real et al., 2017].

Finally, it can be mentioned that deep RL has applications in classic and fundamental algorithmic problems in the field of computer science, such as the travelling salesman problem [Bello et al., 2016]. This is an NP-complete problem and the possibility to tackle it with deep RL shows the potential impact that it could have on all applications that can be cast as NP-complete problems.

2.6.2 *Challenges of applying reinforcement learning to real-world problems*

The deep RL algorithms discussed in Section 2.3 can, in principle, be used to solve many different types of real-world problems. In practice, there is however one fundamental difficulty: it is often not possible to let an agent interact freely and sufficiently in the actual environment (or set of environments). Two main cases may prevent that ideal scenario in real-world applications:

1. The agent may not be able to interact with the true environment but only with a simulation of it. Those constraints may be subsequent to the fact that exploration is severely limited in the actual environment due to safety/cost constraints. For instance, that scenario occurs in smartgrids (see Chapter 6), robotics [Zhu et al., 2016], etc. Note that other alternatives to deal with safety/cost constraints also exist and readers are referred to the following overview: [Garcıa and Fernández, 2015].

2. The acquisition of new observations may not be possible anymore (the "batch" setting, see Section 2.4.1.1). That scenario happens for instance in medical trials, in tasks with dependence on weather conditions or in trading markets (energy markets, stock markets, etc).

Note that a combination of the two scenarios is also possible in the case where the dynamics may be simulated but where the dependence on an exogenous time series is only accessible via limited data. That is the scenario we will discuss in the microgrids setting in Chapter 6.

In order to deal with the 'reality gap' between the simulation and the actual environment (difficulty 1 from above), different elements are important:

- One can aim to develop a simulator that is as accurate as possible.
- One can design the learning algorithm so as to improve generalization and/or use transfer learning methods (see Section 2.4.1).

2.7 OPEN-SOURCE SOFTWARE FOR DEEP RL: DEER

A general schema of the different elements that can be found in most deep RL algorithms (not applicable to inverse RL or imitation learning) is provided in Figure 2.15.

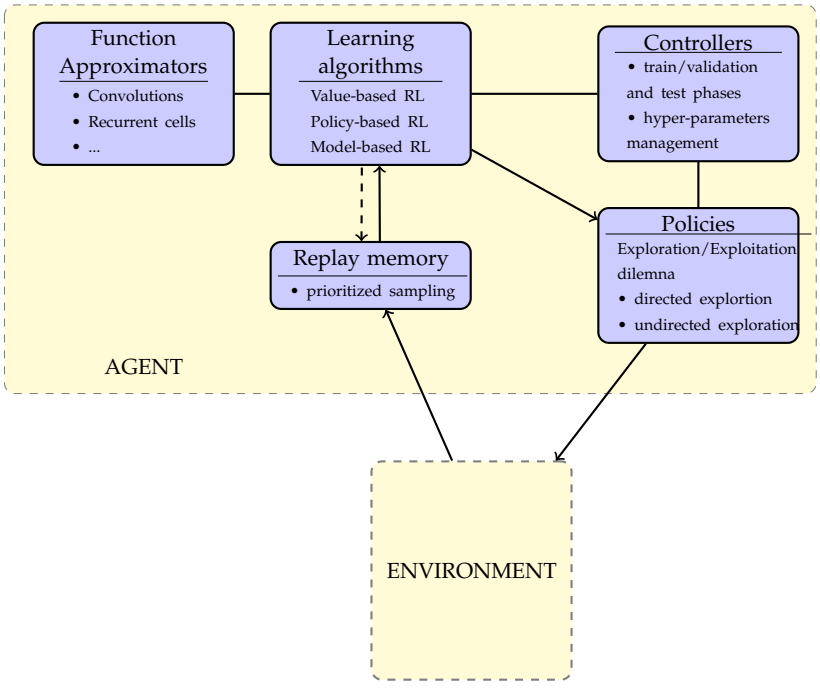


Figure 2.15: General schema of deep RL methods.

Many frameworks already exist for RL. Here is a list of some well-known that use python:

- RL-Glue [Tanner and White, 2009] provides a standard interface that allows to connect RL agents, environments, and experiment programs together.

- PyBrain [Schaul et al., 2010] Machine learning library focused on neural networks with some RL support
- RLPy [Geramifard et al., 2015] is a framework focused on RL based on value-function using linear function approximation with discrete actions.

In the context of this thesis, DeeR has been developed. The library combines out of the box RL and neural networks (similarly to PyBrain). Particular elements unique to the DeeR framework are emphasized here:

- It provides access to policy-based methods and value-based methods that are readily available in the library (with advanced techniques such as Double Q-learning and prioritized experience replay).
- It provides a general framework where the observation is made up of any number of elements: scalars, vectors and frames. The observations history that the agent is based on to build the Q-function or the policy is made up of any truncated history of each element provided in the observation (see Chapter 3 for more information on why it may be interesting to use a truncated history of observations).
- It allows to easily add up a validation phase that allows to stop the training process before overfitting. This possibility is useful when the environment is dependent on scarce data (e.g., limited time series).

The DeeR framework is built on top of the ongoing effort in deep learning to provide an effective open-source eco-system. The objective of DeeR is to maintain an interface that is (1) easily accessible to anyone willing to start using existing algorithms, and (2) efficient and modular for researchers.

The source code is available at `https://github.com/VinF/deer`. The documentation is available at `http://deer.readthedocs.io/`.

2.8 CONCLUSION

The important developments in the field of deep learning have contributed to many new avenues when RL methods and deep learning are combined. In particular, it has been emphasized that deep learning has brought its generalization capabilities making possible to work with large, high-dimensional state and/or action spaces. An overview of the tremendous development in this field both in terms of applications and algorithmic developments has also been provided.

Part I

CONTRIBUTIONS TO DEEP RL

3

BIAS-OVERFITTING IN BATCH REINFORCEMENT LEARNING WITH PARTIAL OBSERVABILITY

3.1 INTRODUCTION

This chapter is dedicated to sequential decision-making problems that may be modeled as Markov Decision Processes for which the system dynamics may be partially observable, a class of problems often called Partially Observable Markov Decision Processes (POMDPs). Within this setting, we focus on decision-making strategies computed using RL. RL approaches rely on observations gathered through interactions with the (PO)MDP, and, although most RL approaches have strong convergence properties, classic RL approaches are challenged by data scarcity. When acquisition of new observations is possible (the "online" case), data scarcity is gradually phased out using strategies balancing the exploration / exploitation (E/E) tradeoff. The scientific literature related to this topic is vast; in particular, Bayesian RL techniques [Ross et al., 2011; Ghavamzadeh et al., 2015] offer an elegant way of formalizing the E/E tradeoff.

However, such E/E strategies are not applicable when the acquisition of new observations is not possible anymore (the "batch" setting). Within this context, we propose to revisit RL as a learning paradigm that faces, similarly to supervised learning, a tradeoff between simultaneously minimizing two sources of error: an asymptotic bias and an overfitting error. The asymptotic bias (also simply called bias in the following) directly relates to the choice of the RL algorithm (and its parameterization). Any RL algorithm defines a policy class as well as a procedure to search within this class, and the bias may be defined as the performance gap between best candidate optimal policies and actual optimal policies. This bias does not depend on the set of observations. On the other hand, overfitting is an error term induced by the fact that only a limited amount of data is available to the algorithm that may potentially overfit suboptimal policies. This overfitting error vanishes as the size and the quality of the dataset increase.

In this chapter, we focus on studying the interactions between these two sources of error, in a setting where the system dynamics is partially observable. Due to this particular setting, one needs to build a state representation from a history of data. By increasing the cardinality of the state representation, the algorithm may be provided with a more informative representation of the POMDP, but at the price of simultaneously increasing the size of the set of candidate policies, thus also increasing the risk of overfitting. We analyze this tradeoff in the case where the RL algorithm provides an optimal solution to the frequentist-based MDP associated with

the state representation (independently of the method used by the learning algorithm to converge towards that solution). Our analysis relies on expressing the quality of a state representation by bounding L_1 error terms of the associated belief states, thus defining ϵ-sufficient statistics in the hidden state dynamics.

Experimental results illustrate the theoretical findings on a distribution of POMDPs in the case where the state representations are truncated histories of observations. In particular, we illustrate the link between the variance observed when dealing with different datasets (directly linked to the size of the dataset) and overfitting, where the link is that variance leads to overfitting if we have a (too) large feature space.

We also discuss and illustrate how using function approximators and adapting the discount factor play a role in the tradeoff between bias and overfitting. This provides the reader with an overview of key elements involved in this tradeoff.

The remainder of the chapter is organized as follows. Section 3.2 formalizes POMDPs, (limited) sets of observations and state representations. Section 3.3 details the main contribution of this chapter: an analysis of the bias-overfitting tradeoff in batch POMDPs. Section 3.4 empirically illustrates the main theoretical results, while Section 3.5 concludes.

3.2 FORMALIZATION

We consider a discrete-time POMDP model M defined as follows:

Definition 3.1 *A POMDP is a 7-tuple* $(\mathcal{S}, \mathcal{A}, T, R, \Omega, O, \gamma)$ *where:*

- $\mathcal{S}$ *is a finite set of states* $\{1, \dots, N_{\mathcal{S}}\}$,
- $\mathcal{A}$ *is a finite set of actions* $\{1, \dots, N_{\mathcal{A}}\}$,
- $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ *is the transition function (set of conditional transition probabilities between states),*
- $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathcal{R}$ *is the reward function, where* $\mathcal{R}$ *is a continuous set of possible rewards in a range* $R_{max} \in \mathbb{R}^+$ *(e.g.,* $[0, R_{max}]$ *without loss of generality),*
- Ω *is a finite set of observations* $\{1, \dots, N_{\Omega}\}$,
- $O : \mathcal{S} \times \Omega \to [0, 1]$ *is a set of conditional observation probabilities, and*
- $\gamma \in [0, 1)$ *is the discount factor.*

The environment starts in a distribution of initial states $b(s_0)$. At each time step $t \in \mathbb{N}_0$, the environment is in a state $s_t \in \mathcal{S}$. At the same time, the agent receives an observation $\omega_t \in \Omega$ which depends

on the state of the environment with probability $O(s_t, \omega_t)$ and the agent has to take an action $a_t \in \mathcal{A}$. Then, the environment transitions to state $s_{t+1} \in \mathcal{S}$ with probability $T(s_t, a_t, s_{t+1})$ and the agent receives a reward $r_t \in \mathcal{R}$ equal to $R(s_t, a_t, s_{t+1})$. In this chapter, the conditional transition probabilities T, the reward function R and the conditional observation probabilities O are unknown (which prevents using methods such as [Pineau et al., 2003]). The only information available to the agent is the past experience gathered while interacting with the POMDP.

3.2.1 *Processing a history of data*

Policies considered in this chapter are mappings from (a set of) observation(s) into actions. In that context, a naive approach to build a space of candidate policies is to consider the set of mappings taking only the very last observation(s) as input. However, in a POMDP setting, this leads to candidate policies that are likely not rich enough to capture the system dynamics, thus suboptimal [Singh et al., 1994; Wolfe, 2006]. there is no alternative to using a history of previously observed features $H_t \in \mathcal{H}$ to better estimate the hidden state dynamics (see Figure 3.1). We denote by $\mathcal{H}_t = \Omega \times (\mathcal{A} \times \mathcal{R} \times \Omega)^t$ the set of histories observed up to time t for $t \in \mathbb{N}_0$, and by $\mathcal{H} = \bigcup_{t=0}^{\infty} \mathcal{H}_t$ the space of all possible observable histories.

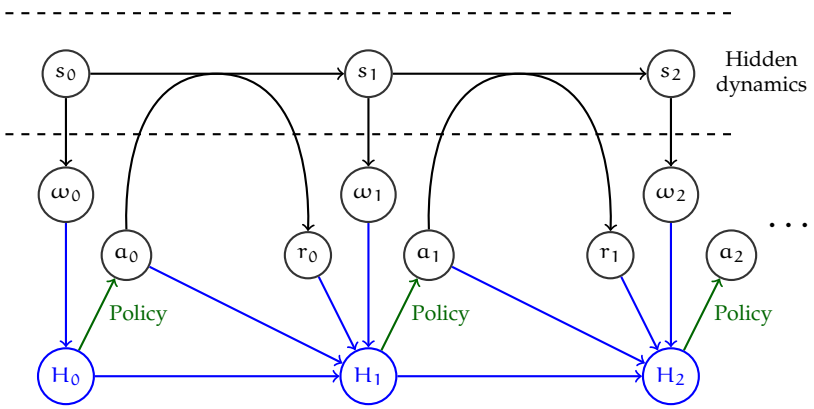


Figure 3.1: Illustration of a POMDP.

A straightforward approach is to take the whole history $H_t \in \mathcal{H}$ as input of candidate policies [Braziunas, 2003]. However, taking a too long history may have several drawbacks. Indeed, increasing the size of the set of candidate optimal policies generally implies: (i) more computation to search within this set [Singh et al., 1994; McCallum, 1996] and (ii) an increased risk of including candidate policies suffering overfitting (see Section 3.3). In this chapter, we are specifically interested in minimizing the latter overfitting drawback while keeping an informative state representation.

We define a mapping $\phi : \mathcal{H} \to \phi(\mathcal{H})$, where $\phi(\mathcal{H}) = \{\phi(H) | H \in \mathcal{H}\}$ is of finite cardinality $|\phi(\mathcal{H})|$. Note that a mapping $\phi(\cdot)$ induces an upper bound on the number of candidate policies: $|\Pi_{\phi(\mathcal{H})}| \leqslant |\mathcal{A}|^{|\phi(\mathcal{H})|}$.

Definition 3.2 *The belief state* $b(s|H_t)$ *(resp.* $b_\phi(s|\phi(H_t))$*) is defined as the vector of probabilities where the* i^{th} *component* ($i \in \{1, \ldots, N_S\}$) *is given by* $\mathbb{P}(s_t = i \mid H_t)$ *(resp.* $\mathbb{P}(s_t = i \mid \phi(H_t))$*), for all sequences* $H_t \in \mathcal{H}$.

Definition 3.3 *A mapping* $\phi_0 : \mathcal{H} \to \phi_0(\mathcal{H})$ *is a particular mapping* ϕ *such that* $\phi_0(H)$ *is a sufficient statistic for the POMDP* M*:*

$$\mathbb{P}(s_t \mid H_t) = \mathbb{P}(s_t \mid \phi_0(H_t)), \tag{3.1}$$

for all states $s_t \in \mathcal{S}$ *and for all sequences* $H_t \in \mathcal{H}$.

Note that $P(s_t|\phi(H))$ is a well-defined probability distribution even if $\phi(H)$ is not a sufficient statistic in the case where we consider a well-defined probability distribution of H.

Definition 3.4 *A mapping* $\phi_\epsilon : \mathcal{H} \to \phi_\epsilon(\mathcal{H})$ *is a particular mapping* ϕ *such that* $\phi_\epsilon(H)$ *is an* ϵ*-sufficient statistic for the POMDP* M *that satisfies the following condition with* $\epsilon \geqslant 0$ *and with the* L_1 *norm:*

$$\|b_{\phi_\epsilon}(\cdot|\phi_\epsilon(H_t)) - b(\cdot|H_t)\|_1 \leqslant \epsilon, \tag{3.2}$$

for all sequences $H_t \in \mathcal{H}$.

3.2.2 *Working with a limited dataset*

Let $\mathcal{M}(\mathcal{S}, \mathcal{A}, \Omega, \gamma)$ be a set of POMDPs with fixed $\mathcal{S}$, $\mathcal{A}$, Ω, and γ. For any $M(T, R, O) \in \mathcal{M}$, we denote by D_{M,π_s,N_{tr},N_l} a random dataset generated according to a probability distribution $\mathcal{D}_{M,\pi_s,N_{tr},N_l}$ over the set of N_{tr} (unordered) trajectories of length N_l. One such trajectory is defined as the observable history $H_{N_l} \in \mathcal{H}_{N_l}$ obtained in M when starting from s_0 and following a stochastic sampling policy π_s that ensures a non-zero probability of taking any action given an observable history $H \in \mathcal{H}$. For simplicity we will denote D_{M,π_s,N_{tr},N_l}, simply as D. For the purpose of the analysis, we also introduce the asymptotic dataset $D_\infty = D_{M,\pi_s,N_{tr}\to\infty,N_l\to\infty}$ that would be theoretically obtained in the case where one could generate an infinite number of observations ($N_{tr} \to \infty$ and $N_l \to \infty$).

In this chapter, the algorithm cannot generate additional data. The challenge is to determine a high-performance policy (in the actual environment) while having only access to a fixed dataset D.

3.2.3 *Assessing the performance of a policy*

In this chapter, we will consider stationary and deterministic control policies $\pi \in \Pi : \phi(\mathcal{H}) \to \mathcal{A}$. Any particular choice of ϕ induces a

particular definition of the policy space Π. We introduce $V^{\pi}_{M}(\phi(H))$ with $H \in \mathcal{H}$ as the expected return obtained over an infinite time horizon when the system is controlled using policy π in the POMDP M:

$$V^{\pi}_{M}(\phi(H)) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^{k} r_{t+k} | \phi(H_t) = \phi(H), \pi, b(s_0)\right], \tag{3.3}$$

where $\mathbb{P}\big(s_{t+1}|s_t, \pi(\phi(H_t))\big) = T(s_t, \pi(\phi(H_t)), s_{t+1})$ and $r_t = R\big(s_t, \pi(\phi(H_t)), s_{t+1}\big)$.

Note that Equation 3.3 is well defined even if $\phi(H)$ is not a sufficient statistic at the condition that $P(s_t|\phi(H))$ is a well-defined probability distribution.

Let π^* be an optimal policy in M defined as:

$$\pi^* \in \underset{\pi:\phi_0(\mathcal{H})\to\mathcal{A}}{\operatorname{argmax}} \; V^{\pi}_{M}(\phi_0(H_0)), \tag{3.4}$$

where H_0 is the distribution of initial observations (compatible with the distribution $b(s_0)$ of initial states through the conditional observation probabilities). Note that, as it will become clear in the following, the definition of π^* is the same as it is usually defined in a POMDP (when known T,R and O and known initial belief state), such as in [Sondik, 1978].

3.3 BIAS-OVERFITTING IN RL WITH PARTIAL OBSERVABILITY

Importance of the feature space

This section introduces and analyzes a bias-overfitting decomposition of the performance gap of RL policies computed from the frequentist-based augmented MDP built from the dataset D. In that setting, the agent behaves optimally with respect to the maximum-likelihood model estimated from the data under the chosen abstraction, which allows removing from the analysis how the RL algorithm converges. Let us first define the frequentist-based augmented MDP:

Definition 3.5 *With M defined by $(S, A, T, R, \Omega, O, \gamma)$ and the dataset D built from interactions with M while following a policy π_s, the frequentist-based augmented MDP $\hat{M}_{D,\phi}$ also denoted for simplicity $\hat{M}_D = (\Sigma, A, \hat{T}, \hat{R}, \Gamma)$ is defined with:*

- *the state space: $\Sigma = \phi(\mathcal{H})$,*
- *the action space: $A = \mathcal{A}$,*
- *the estimated transition function: for $\sigma, \sigma' \in \Sigma$ and $a \in A$, $\hat{T}(\sigma, a, \sigma')$ is the number of times we observe the transition $(\sigma, a) \times \sigma' \to [0, 1]$ in D divided by the number of times we observe (σ, a); if any (σ, a) has never been encountered in a dataset, we set $\hat{T}(\sigma, a, \sigma') = 1/|\Sigma|, \forall \sigma'$,*

- *the estimated reward function: for* $\sigma, \sigma' \in \Sigma$ *and* $a \in A$, $\hat{R}(\sigma, a, \sigma')$ *is the mean of the rewards observed at* (σ, a, σ')*; if any* (σ, a, σ') *has never been encountered in a dataset, we set* $\hat{R}(\sigma, a, \sigma')$ *to the average of rewards observed over the whole dataset D, and*
- *the discount factor* $\Gamma \leqslant \gamma$.

By definition of ϕ_0 and as long as $\phi = \phi_0$, the asymptotic frequentist-based MDP (when unlimited data is available) actually gathers the relevant information from the actual POMDP. Indeed, in the model-based context when the POMDP dynamics is known (i.e., T, R, O), the knowledge of H_t allows calculating the belief state $b(s_t|H_t)$ (calculated recursively thanks to the Bayes rule based on $b(s_t|H_t) = \mathbb{P}(s_t|\omega_t, a_t, b(s_{t-1}|H_{t-1}))$). It is then possible to define, from the history $H \in \mathcal{H}$ and for any action $a \in \mathcal{A}$, the expected immediate reward as well as a transition function into the next observation ω':

- $R_{model-based}(H, a) = \sum_{s' \in \mathcal{S}} \sum_{s \in \mathcal{S}} b(s|H) T(s, a, s') R(s, a, s')$, and
- $T_{model-based}(H, a, \omega') = \sum_{s' \in \mathcal{S}} \underbrace{\sum_{s \in \mathcal{S}} b(s|H) T(s, a, s')}_{\text{next belief state}} O(s', \omega')$.

In the frequentist approach, this information is actually estimated from interactions with the POMDP in $\hat{R}$ and $\hat{T}$ without any explicit knowledge of the dynamics of the POMDP.

We introduce $\mathcal{V}^{\pi}_{\hat{M}_D}(\sigma)$ with $\sigma \in \Sigma$ as the expected return obtained over an infinite time horizon when the system is controlled using a policy π s.t. $a_t = \pi(\sigma_t) : \Sigma \to A, \forall t$ in the augmented decision process $\hat{M}_D$:

$$\mathcal{V}^{\pi}_{\hat{M}_D}(\sigma) = \mathbb{E}\left[\sum_{k=0}^{\infty} \Gamma^k \hat{r}_{t+k} | \sigma_t = \sigma, \pi, b(s_0)\right], \tag{3.5}$$

where $\hat{r}_t$ is a reward s.t. $\hat{r}_t = \hat{R}(\sigma_t, a_t, \sigma_{t+1})$ and the dynamics is given by $\mathbb{P}(\sigma_{t+1}|\sigma_t, a_t) = \hat{T}(\sigma_t, a_t, \sigma_{t+1})$.

Definition 3.6 *The frequentist-based policy* $\pi_{D,\phi}$ *is an optimal policy of the augmented MDP* $\hat{M}_D$ *defined as:* $\pi_{D,\phi} \in \underset{\pi:\Sigma \to A}{\operatorname{argmax}}\ \mathcal{V}^{\pi}_{\hat{M}_D}(\sigma_0)$ *where* $\sigma_0 = \phi(H_0)$.

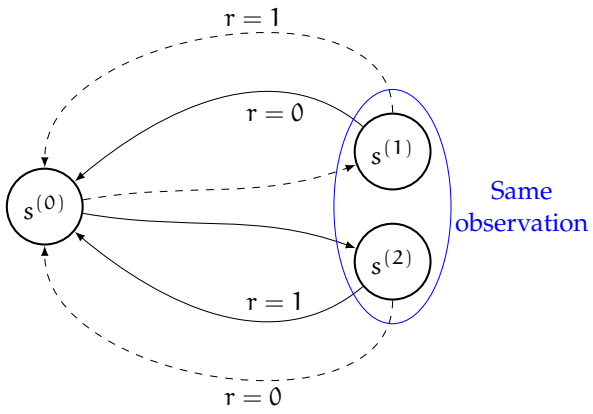


Figure 3.2: Example of a POMDP that illustrates the importance of π_s: at each time step, the agent can choose between action 1 (dashed line) and 2 (solid line). In states $s^{(1)}$ and $s^{(2)}$, the agent obtains the same observation. An agent that would only rely on the current observation (such that it can't discriminate between $s^{(1)}$ or $s^{(2)}$) would have its policy heavily impacted depending on π_s used to build the dataset D because it changes the occupancy distribution of states for each observation. In this case, depending if π_s visits $s^{(1)}$ more often than $s^{(2)}$, the recommended action when visiting those states would be different.

Let us now decompose the error of using a frequentist-based policy $\pi_{D,\phi}$:

$$\mathop{\mathbb{E}}_{D\sim\mathcal{D}}\left[V_M^{\pi^*}(\phi_0(H)) - V_M^{\pi_{D,\phi}}(\phi(H))\right] = \underbrace{\left(V_M^{\pi^*}(\phi_0(H)) - V_M^{\pi_{D_\infty,\phi}}(\phi(H))\right)}_{\substack{\text{bias function of dataset } D_\infty \text{ (function of } \pi_s\text{)} \\ \text{and frequentist-based policy } \pi_{D_\infty,\phi} \text{ (function of } \phi \text{ and } \Gamma\text{)}}} + \underbrace{\mathop{\mathbb{E}}_{D\sim\mathcal{D}}\left[V_M^{\pi_{D_\infty,\phi}}(\phi(H)) - V_M^{\pi_{D,\phi}}(\phi(H))\right]}_{\substack{\text{overfitting due to finite dataset D (function of } \pi_s, N_l, N_{tr}\text{)} \\ \text{in the context of frequentist-based policy } \pi_{D,\phi} \text{ (function of } \phi \text{ and } \Gamma\text{)}}} \tag{3.6}$$

The term *bias* actually refers to an asymptotical bias when the size of the dataset tends to infinity while the term *overfitting* refers to the expected suboptimality due to a finite size of the dataset.

The role of the sampling policy π_s is illustrated in Figure 3.2.

Selecting carefully the feature space $\phi(\mathcal{H})$ allows building a class of policies that have the potential to accurately capture information from data (low bias), but also generalize well (low overfitting). On the one hand, using too many non-informative features will increase overfitting, as stated in Theorem 3.2. On the other hand,

a mapping $\phi(\mathcal{H})$ that discards useful available information will suffer an asymptotic bias, as stated in Theorem 3.1 (arbitrarily large depending on the POMDP and on the features discarded).

Theorem 3.1 *Let M be a POMDP described by the 7-tuple $(\mathcal{S}, \mathcal{A}, T, R, \Omega, O, \gamma)$. Let $\hat{M}_{D_\infty}$ be an augmented MDP $(\Sigma, A, \hat{T}, \hat{R}, \Gamma = \gamma)$ estimated, according to Definition 3.5, from a dataset D_∞. Then the asymptotic bias can be bounded as follows:*

$$\max_{H \in \mathcal{H}} \left(V_M^{\pi^*}(\phi_0(H)) - V_M^{\pi_{D_\infty,\phi}}(\phi(H)) \right) \leqslant \frac{2\epsilon R_{max}}{(1-\gamma)^3}, \tag{3.7}$$

where ϵ is such that the mapping ϕ respects the conditions to be ϕ_ϵ.

The proof is deferred to Appendix 3.6.1. This bound is an original result based on the belief states (which was not considered in other works) via the ϵ-sufficient statistic (L_1 norm error). Note that bisimulation metrics ([Ferns et al., 2004]) may also be used to take into account how the errors on the belief states may have less of an impact at the condition that the hidden states affected by these errors are close according to the bisimulation metrics.

We now provide a bound on the overfitting error that monotonically grows with $|\phi(\mathcal{H})|$.

Theorem 3.2 *Let M be a POMDP described by the 7-tuple $(\mathcal{S}, \mathcal{A}, T, R, \Omega, O, \gamma)$. Let $\hat{M}_D$ be an augmented MDP $(\Sigma, A, \hat{T}, \hat{R}, \Gamma = \gamma)$ estimated, according to Definition 3.5, from a dataset D with the assumption that D has n transitions from any possible pair $(\phi(H), a) \in (\phi(\mathcal{H}), A)$. Then the overfitting due to using the frequentist-based policy $\pi_{D,\phi}$ instead of $\pi_{D_\infty,\phi}$ in the PODMP M can be bounded as follows:*

$$\max_{H \in \mathcal{H}} \left(V_M^{\pi_{D_\infty,\phi}}(\phi(H)) - V_M^{\pi_{D,\phi}}(\phi(H)) \right) \leqslant \frac{2R_{max}}{(1-\gamma)^2} \sqrt{\frac{1}{2n} \ln \left(\frac{2|\phi(\mathcal{H})||A|^{1+|\phi(\mathcal{H})|}}{\delta} \right)}, \tag{3.8}$$

with probability at least $1-\delta$.

The proof is deferred to Appendix 3.6.2. Theorem 3.2 shows that using a large set of features allows a larger policy class, hence potentially leading to a stronger drop in performance when the available dataset D is limited (the bound decreases proportionally to $\frac{1}{\sqrt{n}}$). A theoretical analysis in the context of MDPs with a finite dataset was performed in [Jiang et al., 2015a].

Overall Theorems 3.1 and 3.2 can help to choose a good state representation for POMDPs as they provide bounds on the two terms that appear in the bias-overfitting decomposition of Equation 3.6. For example, an additional feature in the mapping ϕ has an overall positive effect only if it provides a significant increase of information on the belief state (i.e., if it allows obtaining a more

accurate knowledge of the underlying state of the MDP defined by T and R when given $\phi(H)$). This increase of information must be significant enough to compensate for the additional risk of overfitting when choosing a large cardinality of $\phi(H)$. Note that one could combine the two bounds to theoretically define an optimal choice of the state representation with lower bound guarantees regarding the bias-overfitting tradeoff.

However, as the two bounds are loose, it is not at all assured that a general and efficient algorithm can result directly from these bounds in practice. Here are two examples that illustrate how these bounds will be more or less loose depending on the problem:

- One can consider a problem where a given ϕ_ϵ provides in fact sufficient statistics in a large part of the history space $\mathcal{H}$. In that case, the bound on the bias would a priori be very loose.
- One can consider a problem where one or several of the features given by $\phi(\mathcal{H})$ are duplicates of other features (or even linear combinations of other features). In that case, the overfitting bound will be higher due to these additional features, even though the actual risk of overfitting does not increase as compared to the problem where these features are removed.

Importance of function approximators

As described earlier, a straightforward mapping $\phi(\cdot)$ may be obtained by discarding a few features from the observable history. However, it is often more efficient to rely on a smarter processing of the features by selecting a suitable function approximator structure (e.g., in a Q-learning scheme) that constrains policies to have interesting generalization properties. This analysis can be captured in a theorem similar to Theorem 3.2 which takes into account the complexity of the function approximator and uses the Rademacher complexity measure in order to provide a bound potentially much tighter:

Theorem 3.3 *Let M be a POMDP described by the 7-tuple $(S, \mathcal{A}, T, R, \Omega, O, \gamma)$. Let $\hat{M}_D$ be an augmented MDP $(\Sigma, A, \hat{T}, \hat{R}, \Gamma = \gamma)$ estimated, according to Definition 3.5, from a dataset D with the assumption that D has n transitions from any possible pair $(\phi(H), a)$, $a \in A$. Then the overfitting due to using the frequentist-based policy $\pi_{D,\phi}$ instead of $\pi_{D_\infty,\phi}$ in the PODMP M can be bounded as follows:*

$$\begin{aligned}
&\max_{H\in\mathcal{H}} \left(V_M^{\pi_{D_\infty,\phi}}(\phi(H)) - V_M^{\pi_{D,\phi}}(\phi(H))\right) \\
&\leqslant \frac{2}{1-\gamma}\left(2 \max_{\varphi\in\phi(\mathcal{H}),a\in A} \hat{\mathfrak{R}}_{D_{\varphi,a}}(\mathcal{F}_{M,\phi}) \right. \\
&\qquad\qquad \left. + 3\frac{R_{max}}{1-\gamma}\sqrt{\frac{1}{2n}\ln\left(\frac{4|\phi(\mathcal{H})||A|}{\delta}\right)}\right), \quad (3.9)
\end{aligned}$$

with probability at least $1-\delta$, where

- $\mathcal{F}_{M,\phi} = \{f^{\pi}_{M,\phi} : \pi \in \Phi(\mathcal{H}) \to A\}$, *with* $f^{\pi}_{M,\phi}(r, \varphi') = r + \gamma V^{\pi}_{M}(\varphi')$, *and*
- $\hat{\mathfrak{R}}_{D_{\varphi,a}}(\mathcal{F}_{M,\phi})$ *is the empirical Rademacher complexity of the function class* $\mathcal{F}_{M,\phi}$ *with respect to input points* $D_{\varphi,a}$*:*

$$\mathop{\mathbb{E}}_{\substack{\sigma_i \sim unif\{-1,1\} \\ i=1,\dots,n}} \left\{ \sup_{f \in \mathcal{F}_{M,\phi}} \frac{1}{n} \sum_{(r,\varphi') \in D_{\varphi,a}} \sigma_i f(r, \varphi') \right\}$$

where $D_{\varphi,a}$ *is the set of* n *pairs of immediate reward* r *and next-state representation* φ' *sampled from* (φ, a) *in dataset* D*.*

The proof is deferred to Appendix 3.6.3.

It is worth noting that in the case of neural networks, architectures such as convolutional layers or recurrency are particularly well-suited to deal with a large input space because they offer interesting generalization properties that are adapted to high-dimensional sensory inputs for which hierarchical patterns can be found [LeCun et al., 2015]. A few recent successes make use of convolutional layers [Mnih et al., 2015] and/or recurrent layers [Hausknecht and Stone, 2015] (e.g., LSTMs [Hochreiter and Schmidhuber, 1997]) to solve large scale POMDPs.

Importance of the discount factor Γ *used in the training phase:*

Artificially lowering the discount factor has been shown to improve the performance of the policy when solving MDPs with limited data [Petrik and Scherrer, 2009; Jiang et al., 2015b]. In the partially observable setting, these results may be transferred to the frequentist-based MDP $(\Sigma, A, \hat{T}, \hat{R}, \Gamma)$.

Selection of the parameters with validation or cross-validation to balance the bias-overfitting tradeoff

In the batch setting case, the selection of the policy parameters to effectively balance the bias-overfitting tradeoff can be done similarly to that in supervised learning (e.g., cross-validation) as long as the performance criterion can be estimated from a subset of the trajectories from the dataset D not used during training (validation set). One possibility is to fit an MDP model from data via the frequentist approach (or regression), and evaluate the policy against the model. Another approach is to use the idea of importance sampling [Precup, 2000]. A mix of the two approaches can be found in [Jiang and Li, 2016; Thomas and Brunskill, 2016]. Another approach is to perform a policy evaluation based solely on a set of tuples with techniques based on Lipschitz continuity assumptions on the system dynamics, which are in described in [Fonteneau et al., 2010].

Note that there exists a particular case where the system is modeled through a dynamics that is available to the agent but with

a dependence on an exogenous time series (e.g., trading, weather-dependent dynamics) for which the agent only has finite data. In that case, the exogenous signal can be broken down so as to allow a validation strategy, as done in Chapter 6.

3.4 EXPERIMENTS

3.4.1 *Protocol*

We randomly sample N_P POMDPs such that $N_{\mathcal{S}} = 5$, $N_{\mathcal{A}} = 2$ and $N_{\Omega} = 5$ (except when stated otherwise) from a distribution $\mathcal{P}$ that we refer to as Random POMDP. The distribution $\mathcal{P}$ is fully determined by specifying a distribution over the set of possible transition functions $T(\cdot,\cdot,\cdot)$, a distribution over the set of reward functions $R(\cdot,\cdot,\cdot)$, and a distribution over the set of possible conditional observation probabilities $O(\cdot,\cdot)$.

Random transition functions $T(\cdot,\cdot,\cdot)$ are drawn by assigning, for each entry (s,a,s'), a zero value with probability $3/4$, and, with probability $1/4$, a non-zero entry with a probability drawn uniformly in $[0,1]$. For all (s,a), if all $T(s,a,s')$ are zeros, we enforce one non-zero value for a random $s' \in \mathcal{S}$. Values are normalized.

Random reward functions are generated by associating to all possible (s,a,s') a reward sampled uniformly and independently from $[-1,1]$.

Random conditional observation probabilities $O(\cdot,\cdot)$ are generated the following way: the probability to observe $o^{(i)}$ when being in state $s^{(i)}$ is equal to 0.5, while all other values are chosen uniformly randomly so that it is normalized for any s.

For all POMDPs, we fixed $\gamma = 1$ and $\Gamma = 0.95$ if not stated otherwise and we truncate the trajectories to a length of $N_l = 100$ time steps.

For each generated POMDP $P \sim \mathcal{P}$, we generate 20 datasets $D \in \mathcal{D}_P$ where $\mathcal{D}_P$ is a probability distribution over all possible sets of n trajectories ($n \in [2, 5000]$); where each trajectory is made up of an history H_{100} of 100 time steps, when starting from an initial state $s_0 \in \mathcal{S}$ while taking uniformly random decisions. Each dataset D induces a policy $\pi_{D,\phi}$, and we want to evaluate the expected return of this policy while discarding the variance related to the stochasticity of the transitions, observations and rewards. To do so, policies are tested with 1000 rollouts of the policy. For each POMDP P, we are then able to get an estimate of the average score μ_P which is defined as:

$$\mu_P = \mathop{\mathbb{E}}_{D\sim\mathcal{D}_P} \mathop{\mathbb{E}}_{\text{rollouts}} \left[\sum_{t=0}^{N_l} \gamma^t r_t | s_0, \pi_{D,\phi} \right]. \tag{3.10}$$

We are also able to get an estimate of a parametric variance σ_P^2 defined as:

$$\sigma_P^2 = \underset{D\sim\mathcal{D}_P}{\mathrm{var}} \underset{\text{rollouts}}{\mathbb{E}} \left[\sum_{t=0}^{N_l} \gamma^t r_t | s_0, \pi_{D,\phi} \right]. \tag{3.11}$$

3.4.2 *History processing*

In this section, we show experimentally that any additional feature is likely to reduce the asymptotic bias, but may also increase the overfitting. For each dataset D, we define the policy $\pi_{D,\phi}$ according to Definition 3.6 for every history of interest $h \in \{1,2,3\}$. When $h = 1$ (resp. $h = 2$) (resp. $h = 3$), only the current observation (resp. the last two observations as well as the last action) (resp. the three last observations and the two last actions) is (resp. are) used for building the state of the frequentist-based augmented MDP.

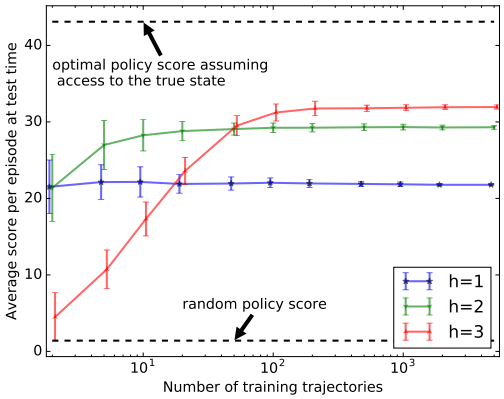


Figure 3.3: Evolution (as a function of the size of the dataset) of estimated values of $\underset{P\sim\mathcal{P}}{\mathbb{E}}\mu_P \pm \underset{P\sim\mathcal{P}}{\mathbb{E}}\sigma_P$ computed from a sample of $N_P = 50$ POMDPs drawn from $\mathcal{P}$. The bars are used to represent the variance observed when dealing with different datasets drawn from a distribution; note that this is not a usual error bar.

The values $\underset{P\sim\mathcal{P}}{\mathbb{E}}\mu_P$ and $\underset{P\sim\mathcal{P}}{\mathbb{E}}\sigma_P$ are displayed in Figure 3.3. One can observe that a small set of features (small history) appears to be a better choice (in terms of total bias) when the dataset is poor (only a few trajectories). With an increasing number of trajectories, the optimal choice in terms of number of features ($h = 1, 2$ or 3) also increases. In addition, one can also observe that the expected variance of the score decreases as the number of samples increases. As the variance decreases, the risk of overfitting also decreases, and it becomes possible to target a larger policy class (using a larger feature set).

The overfitting error is also linked to the variance of the value function estimates in the frequentist-based MDP. When these estimates

have a large variance, an overfitting term appears because of a higher chance of picking one of the suboptimal policies (when using Definition 3.6), as illustrated in Figure 3.4. Note that this variance term has already been studied in the context of estimating value functions in finite MDPs using frequentist approximations of the parameters of the MDPs ($\hat{R}$ and $\hat{T}$) [Mannor et al., 2007].

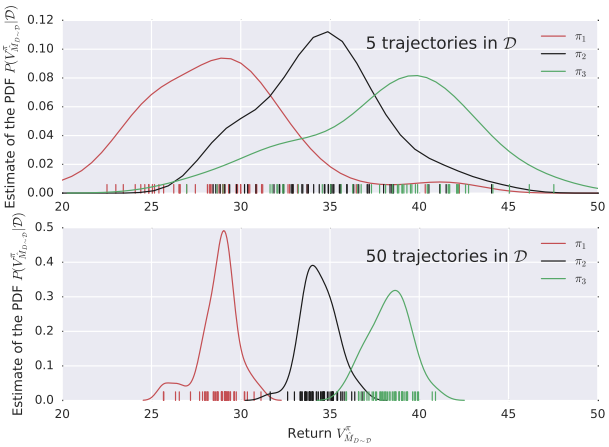


Figure 3.4: Illustrations of the return estimates in the augmented MDP $\hat{M}_D$ ($h = 2$) for three different policies. The policies are selected specifically for illustration purpose based on the criterion $V_M^{\pi_D}$; the best performing (in green), the worst performing (in red) and the median performing were selected in a set of 50 policies built when $D \sim \mathcal{D}$ has 5 trajectories of data from the actual POMDP. On the actual POMDP, the expected returns are $V_M^{\pi_1} = 28$, $V_M^{\pi_2} = 33$, $V_M^{\pi_3} = 42$ (in general, these values need not be the same as the expected value of the probability distribution in the two graphs).

3.4.3 *Function approximator and discount factor*

We also illustrate the effect of using function approximators on the bias-overfitting tradeoff. To do so, we process the output of the state representation $\phi(\cdot)$ into a deep Q-learning scheme (technical details are given in appendix 3.6.4). We can see in Figure 3.5 that Deep Q-learning policies suffer less overfitting (better performance in the low data regime) while avoiding introducing an important asymptotic bias (identical performance when lots of data is available). Note that the variance is slightly larger than in Figure 3.3, and does not vanish to 0 with additional data. This is due to the additional stochasticity induced when building the Q-value function with neural networks (note that when performing the same experiments while taking the average recommendation of several Q-value functions, this variance decreases with the number of Q-value functions).

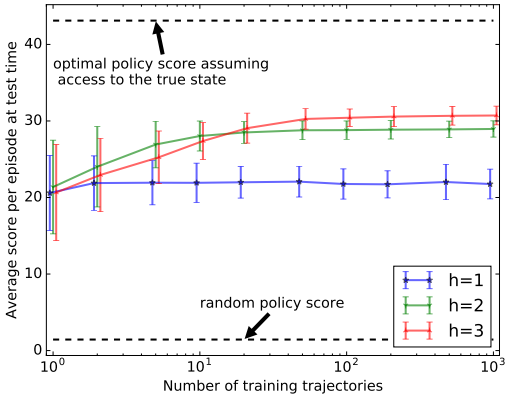


Figure 3.5: Evolution (as a function of the size of the dataset) of estimated values of $\mathop{\mathbb{E}}_{P\sim\mathcal{P}} \mu_P \pm \mathop{\mathbb{E}}_{P\sim\mathcal{P}} \sigma_P$ computed from a sample of $N_P = 50$ POMDPs drawn from $\mathcal{P}$ (same as Figure 3.3) with neural network as a function approximator. The bars are used to represent the variance observed when dealing with different datasets drawn from a distribution; note that this is not a usual error bar.

Finally, we empirically illustrate in Figure 3.6 the effect of the discount factor. When a training discount factor is lower than the one used in the actual POMDP ($\Gamma < \gamma$), there is an additional bias term, while when a high discount factor is used with a limited amount of data, overfitting increases. In our experiments, the influence of the discount factor is more subtle as compared to the impact of the state representation and the function approximator. The influence is nonetheless clear: it is better to have a low discount factor when only a few data is available, and it is better to have a high discount factor when a lot of data is available.

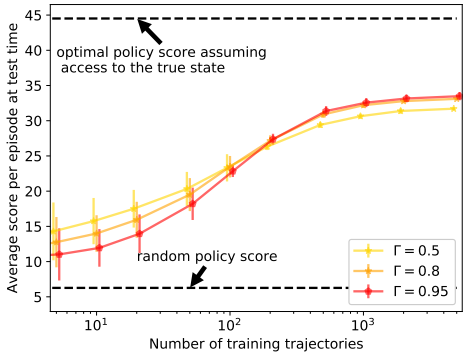


Figure 3.6: Evolution in the frequentist-based case (as a function of the size of the dataset) of estimated values of $\mathop{\mathbb{E}}_{P\sim\mathcal{P}} \mu_P \pm \mathop{\mathbb{E}}_{P\sim\mathcal{P}} \sigma_P$ computed from a sample of $N_P = 10$ POMDPs drawn from $\mathcal{P}$ with $N_S = 8$ and $N_\Omega = 8$ ($h = 3$). The bars are used to represent the variance observed when dealing with different datasets drawn from a distribution; note that this is not a usual error bar.

3.5 CONCLUSION AND FUTURE WORKS

This chapter discusses the bias-overfitting tradeoff of batch RL algorithms in the context of POMDPs. We propose an analysis showing that, similarly to supervised learning techniques, RL may face a bias-overfitting dilemma in situations where the policy class is too large compared to the batch of data. In such situations, we show that it may be preferable to concede an asymptotic bias in order to reduce overfitting. This (favorable) asymptotic bias may be introduced through different manners: (i) downsizing the state representation, (ii) using specific types of function approximators and (iii) lowering the discount factor.

The originality of the setting proposed in this chapter compared to [Maillard et al., 2011; Ortner et al., 2014] and the related work is mainly to formalize the problem in a batch setting (limited set of tuples) instead of the online setting. As compared to [Jiang et al., 2015b], the originality is to consider a partially observable setting.

The work proposed in this chapter may also be of interest in online settings because at each stage, obtaining a performant policy from given data is part of the solution to an efficient exploration/-exploitation tradeoff. For instance, this sheds lights on the interest of progressively increasing the discount factor (which will be discussed in the next chapter). Besides, optimizing the bias-overfitting tradeoff suggests to dynamically (along learning) adapt the feature space and the function approximator (e.g., through ad hoc regularization, or by adapting the neural network architecture, using for instance the NET2NET transformation [Chen et al., 2015]).

3.6 APPENDIX

3.6.1 *Proof of Theorem 3.1*

SKETCH OF THE PROOF

The idea of this proof is to consider two different histories that lead to the same ϵ-sufficient statistic. We then show that the belief state of these two histories (knowing the full histories or the sufficient statitics) is bounded in L_1 norm linearly with ϵ (which is subsequent to our definition of the ϵ-sufficient statistics). In that context we can calculate a bound on the optimal Q-value function when faced with those two histories. Using the Lemma 1 from [Abel et al., 2016], the theorem follows.

In the frequentist-based MDP $\hat{M}_{D_\infty,\phi_0}(\Sigma_0, A, \hat{T}, \hat{R}, \Gamma = \gamma)$, for $H \in \mathcal{H}$, let us define

$$Q^{\pi_{D_\infty,\phi_0}}_{\hat{M}_{D_\infty,\phi_0}}(\phi_0(H), a) = \hat{R}'(\phi_0(H), a) + \gamma \sum_{\varphi \in \phi_0(\mathcal{H})} \hat{T}(\phi_0(H), a, \varphi) V^{\pi_{D_\infty,\phi_0}}_{\hat{M}_{D_\infty,\phi_0}}(\varphi)$$

where $\hat{R}'(\phi_0(H), a) = \sum_{\varphi \in \phi_0(\mathcal{H})} \hat{T}(\phi_0(H), a, \varphi)\hat{R}(\phi_0(H), a, \varphi)$. Then $\forall H^{(1)}, H^{(2)} \in \mathcal{H}$: $\phi_\epsilon(H^{(1)}) = \phi_\epsilon(H^{(2)})$, it follows that

$$\begin{aligned}
&\max_{H^{(1)},H^{(2)},a} \left| Q^{\pi_{D_\infty,\phi_0}}_{\hat{M}_{D_\infty,\phi_0}}(\phi_0(H^{(1)}), a) - Q^{\pi_{D_\infty,\phi_0}}_{\hat{M}_{D_\infty,\phi_0}}(\phi_0(H^{(2)}), a) \right| \\
&\quad = \max_{H^{(1)},H^{(2)},a} \left| \hat{R}'(\phi_0(H^{(1)}), a) - \hat{R}'(\phi_0(H^{(2)}), a) + \gamma \right. \\
&\quad \left. \sum_{\varphi \in \phi_0(\mathcal{H})} \left(\hat{T}(\phi_0(H^{(1)}), a, \varphi) - \hat{T}(\phi_0(H^{(2)}), a, \varphi) \right) V^{\pi_{D_\infty,\phi_0}}_{\hat{M}_{D_\infty,\phi_0}}(\varphi) \right|
\end{aligned}$$

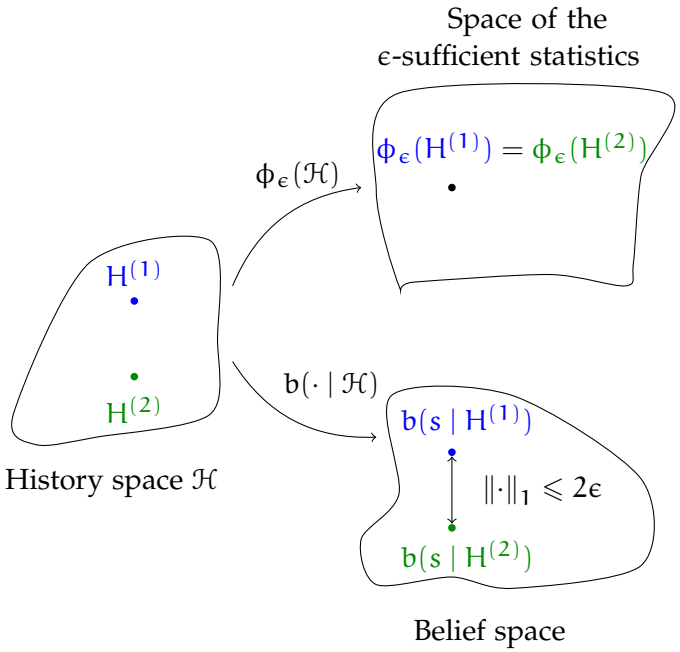


Figure 3.7: Illustration of the ϕ_ϵ mapping and the belief for $H^{(1)}, H^{(2)} \in \mathcal{H}$: $\phi_\epsilon(H^{(1)}) = \phi_\epsilon(H^{(2)})$.

As illustrated in Figure 3.7 and by using the definition of the ϵ-sufficient statistics, we have

$$\begin{aligned}
\left\| b(\cdot \mid H^{(1)}) - b(\cdot \mid H^{(2)}) \right\|_1 &= \left\| b(\cdot \mid H^{(1)}) - b_{\phi_\epsilon}\left(\cdot \mid \phi_\epsilon(H^{(1)})\right) \right. \\
&\quad \left. + b_{\phi_\epsilon}\left(\cdot \mid \phi_\epsilon(H^{(2)})\right) - b(\cdot \mid H^{(2)}) \right\|_1 \\
&\leqslant \left\| b(\cdot \mid H^{(1)}) - b_{\phi_\epsilon}\left(\cdot \mid \phi_\epsilon(H^{(1)})\right) \right\|_1 \\
&\quad + \left\| b_{\phi_\epsilon}\left(\cdot \mid \phi_\epsilon(H^{(2)})\right) - b(\cdot \mid H^{(2)}) \right\|_1 \\
&\leqslant 2\epsilon,
\end{aligned}$$

which has the consequence that

$$\max_{H^{(1)},H^{(2)},a} \left|\hat{R}'(\phi_0(H^{(1)}),a) - \hat{R}'(\phi_0(H^{(2)}),a)\right| \leqslant \epsilon R_{max}$$

It follows that:

$$\begin{aligned}
&\max_{H^{(1)},H^{(2)},a} \left|Q^{\pi_{D_\infty,\phi_0}}_{\hat{M}_{D_\infty,\phi_0}}(\phi_0(H^{(1)}),a) - Q^{\pi_{D_\infty,\phi_0}}_{\hat{M}_{D_\infty,\phi_0}}(\phi_0(H^{(2)}),a)\right| \\
&\quad\leqslant \epsilon R_{max} + \gamma \max_{H^{(1)},H^{(2)},a} \\
&\quad\quad \sum_{\varphi\in\phi_0(\mathcal{H})} \left|\left(\hat{T}(\phi_0(H^{(1)}),a,\varphi) - \hat{T}(\phi_0(H^{(2)}),a,\varphi)\right) V^{\pi_{D_\infty,\phi_0}}_{\hat{M}_{D_\infty,\phi_0}}(\varphi)\right| \\
&\quad= \epsilon R_{max} + \gamma\epsilon\frac{R_{max}}{1-\gamma} = \epsilon\frac{R_{max}}{1-\gamma}
\end{aligned}$$

where the last line is obtained by noticing that the transition functions are always normalized such that

$$\sum_{\varphi\in\phi_0(\mathcal{H})} \left(\hat{T}(\phi_0(H^{(1)}),a,\varphi) - \hat{T}(\phi_0(H^{(2)}),a,\varphi)\right) = 0$$

and that

$$0 \leqslant \max_{\varphi\in\phi_0(\mathcal{H})} \left|V^{\pi_{D_\infty,\phi_0}}_{\hat{M}_{D_\infty,\phi_0}}(\varphi)\right| \leqslant \frac{R_{max}}{1-\gamma},$$

with $\mathcal{R} \in [0, R_{max}]$.

Applying Lemma 1 from [Abel et al., 2016], we obtain:

$$\left\|V^{\pi^*_{D_\infty,\phi_0}}_{M_{D_\infty,\phi_0}} - V^{\pi^*_{D_\infty,\phi_\epsilon}}_{M_{D_\infty,\phi_0}}\right\|_\infty \leqslant \frac{2\frac{\epsilon R_{max}}{1-\gamma}}{(1-\gamma)^2} = \frac{2\epsilon R_{max}}{(1-\gamma)^3}.$$

By further noticing that the dynamics provided by $\hat{M}_{D_\infty,\phi_0}$ and M when starting in s_0 provide an identical value function for a given policy $\pi_{D,\phi}$ and that $\pi_{D_\infty,\phi_0} \sim \pi^*$, thus $V^{\pi^*}_M = V^{\pi_{D_\infty,\phi_0}}_M$, the theorem follows.

3.6.2 *Proof of Theorem 3.2*

SKETCH OF THE PROOF

The idea of the proof is first to bound the difference between value functions following different policies by a bound between value functions estimated in different environments but following the same policy (more precisely a max over a set of policies of such a bound). Once that is done, a bound in probability using Hoeffding's inequality can be obtained.

Let us denote $\varphi \in \phi(\mathcal{H})$:

$$\begin{aligned} V_M^{\pi_{D_\infty,\phi}}(\varphi) - V_M^{\pi_{D,\phi}}(\varphi) =& (V_M^{\pi_{D_\infty,\phi}}(\varphi) - \mathcal{V}_{\hat{M}_D}^{\pi_{D_\infty,\phi}}(\varphi)) \\ & - (V_M^{\pi_{D,\phi}}(\varphi) - \mathcal{V}_{\hat{M}_D}^{\pi_{D,\phi}}(\varphi)) \\ & + (\mathcal{V}_{\hat{M}_D}^{\pi_{D_\infty,\phi}}(\varphi) - \mathcal{V}_{\hat{M}_D}^{\pi_{D,\phi}}(\varphi)) \\ \leqslant & (V_M^{\pi_{D_\infty,\phi}}(\varphi) - \mathcal{V}_{\hat{M}_D}^{\pi_{D_\infty,\phi}}(\varphi)) \\ & - (V_M^{\pi_{D,\phi}}(\varphi) - \mathcal{V}_{\hat{M}_D}^{\pi_{D,\phi}}(\varphi)) \\ \leqslant & 2 \max_{\pi \in \{\pi_{D_\infty,\phi}, \pi_{D,\phi}\}} \left| V_M^{\pi}(\varphi) - \mathcal{V}_{\hat{M}_D}^{\pi}(\varphi) \right|. \end{aligned}$$

It follows that $\forall H$,

$$\begin{aligned} & V_M^{\pi_{D_\infty,\phi}}(\varphi) - V_M^{\pi_{D,\phi}}(\varphi) \\ & \qquad \leqslant 2 \max_{\pi \in \{\pi_{D_\infty,\phi}, \pi_{D,\phi}\}} \max_{\varphi \in \phi(\mathcal{H})} \left| Q_M^{\pi}(\varphi, \pi(\varphi)) - \mathcal{Q}_{\hat{M}_D}^{\pi}(\varphi, \pi(\varphi)) \right| \\ & \qquad \leqslant 2 \max_{\pi \in \{\pi_{D_\infty,\phi}, \pi_{D,\phi}\}} \max_{\varphi \in \phi(\mathcal{H}), a \in A} \left| Q_M^{\pi}(\varphi, a) - \mathcal{Q}_{\hat{M}_D}^{\pi}(\varphi, a) \right|, \end{aligned}$$

where $Q_M^{\pi}(\varphi, a)$ is the action-value function for policy π in M with $a \in \mathcal{A}$. Similarly $\mathcal{Q}_{\hat{M}}^{\pi}(\varphi, a)$ is the action-value function for policy π in $\hat{M}$.

By Lemma 1, we have:

$$\begin{aligned} V_M^{\pi_{D_\infty,\phi}}(\varphi) - V_M^{\pi_{D,\phi}}(\varphi) \leqslant \frac{2}{1-\gamma} & \max_{\pi \in \{\pi_{D_\infty,\phi}, \pi_{D,\phi}\}} \max_{\varphi \in \phi(\mathcal{H}), a \in \mathcal{A}} \\ & \left| \hat{R}'(\varphi, a) + \gamma \sum_{\varphi' \in \phi(\mathcal{H})} \hat{T}(\varphi, a, \varphi') V_M^{\pi}(\varphi') - Q_M^{\pi}(\varphi, a) \right|. \end{aligned} \tag{3.12}$$

where $\hat{R}'(\phi_0(H), a) = \sum_{\varphi \in \phi_0(\mathcal{H})} \hat{T}(\phi_0(H), a, \varphi) \hat{R}(\phi_0(H), a, \varphi)$.

The right-hand side of that equation can be bounded in probability by using Hoeffding's inequality. An adversarial setting is illustrated in Figure 3.8.

With $\mathcal{R} \in [0, R_{max}]$, we notice that $\hat{R}'(\varphi, a) + \gamma \sum_{\varphi' \in \phi(\mathcal{H})} \hat{T}(\varphi, a, \varphi') V_M^{\pi}(\varphi')$ is the mean of i.i.d. variables bounded in the interval $[0, \frac{R_{max}}{1-\gamma}]$ and with mean $Q_M^{\pi}(\varphi, a)$ for any policy $\pi : \phi(\mathcal{H}) \to A$. Therefore, according to Hoeffding's inequality [Hoeffding, 1963], we have with n the number of tuples for every pair (φ, a):

$$\begin{aligned} \mathbb{P} \left\{ \left| \hat{R}'(\varphi, a) + \gamma \sum_{\varphi' \in \phi(\mathcal{H})} \hat{T}(\varphi, a, \varphi') V_M^{\pi}(\varphi') - Q_M^{\pi}(\varphi, a) \right| > t \right\} & \\ \leqslant 2 \exp \left(\frac{-2nt^2}{(R_{max}/(1-\gamma))^2} \right). & \end{aligned} \tag{3.13}$$

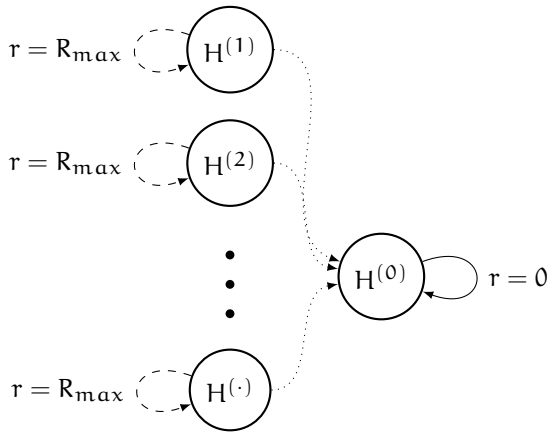


Figure 3.8: Example of a POMDP with an adversarial setting for the overfitting bound (only one action is represented here). Dotted and dashed lines represent transition probabilities p with $0 < p < 1$ and solid lines represent a transition probability 1.

As we want to obtain a bound over all pairs (φ, a) and a union bound on all policies $\pi \in \Pi$ s.t. $|\Pi| = |A|^{|\phi(\mathcal{H})|}$ (indeed Equation 3.13 does not hold for $\pi_{D,\phi}$ alone because that policy is not chosen randomly), we want the right-hand side of equation 3.13 to be $\frac{\delta}{|\Pi||\phi(\mathcal{H})||A|}$. This gives $t(\delta) = \left(\frac{R_{max}}{1-\gamma}\right)\sqrt{\left(\frac{1}{2n}\ln\left(\frac{2|\Pi||\phi(\mathcal{H})||A|}{\delta}\right)\right)}$ and we conclude that:

$$\max_{\varphi\in\phi(\mathcal{H})}\left(V_M^{\pi_{D_\infty,\phi}}(\varphi) - V_M^{\pi_{D,\phi}}(\varphi)\right) \leqslant \frac{2R_{max}}{(1-\gamma)^2}\sqrt{\frac{1}{2n}\ln\left(\frac{2|\phi(\mathcal{H})||A|^{1+|\phi(\mathcal{H})|}}{\delta}\right)},$$

with probability at least $1-\delta$.

LEMMA 1 For any $M =< S, A, T, R, \Omega, O, \gamma >$ and the frequentist-based augmented MDP $\hat{M} =< \Sigma, A, \hat{T}, \hat{R}, \Gamma >$ defined from M according to definition 3.5, we have $\forall \pi : \phi(\mathcal{H}) \to A$ that

$$\max_{\varphi\in\phi(\mathcal{H}),a\in A}\left|Q_M^\pi(\varphi,a) - Q_{\hat{M}_D}^\pi(\varphi,a)\right| \leqslant \frac{1}{1-\gamma}\max_{\varphi\in\phi(\mathcal{H}),a\in A}\left|\hat{R}'(\varphi,a) + \gamma\sum_{\varphi'\in\phi(\mathcal{H})}\hat{T}(\varphi,a,\varphi')V_M^\pi(\varphi') - Q_M^\pi(\varphi,a)\right|. \quad (3.14)$$

PROOF OF LEMMA 1 Given any policy π, let us define $Q_0, Q_1, ..., Q_m$ s.t.

- $Q_0(\varphi,a) = Q_M^\pi(\varphi,a)$ and

- $Q_m(\varphi, a) = \hat{R}'(\varphi, a) + \gamma \sum_{\varphi' \in \phi(\mathcal{H})} \hat{T}(\varphi, a, \varphi') V_{m-1}(\varphi')$,

where $V_{m-1}(\varphi) = Q_{m-1}(\varphi, \pi(\varphi))$. Therefore, we have

$$\begin{aligned}
&\|Q_m - Q_{m-1}\|_\infty \\
&\quad \leqslant \gamma \max_{\varphi \in \phi(\mathcal{H}), a \in A} \left| \sum_{\varphi' \in \phi(\mathcal{H})} \hat{T}(\varphi, a, \varphi')(V_{m-1} - V_{m-2})(\varphi') \right| \\
&\quad \leqslant \gamma \|(V_{m-1} - V_{m-2})(\varphi)\|_\infty \\
&\quad \leqslant \gamma \|(Q_{m-1} - Q_{m-2})(\varphi, a)\|_\infty .
\end{aligned}$$

Taking the limit of $m \to \infty, Q_m \to Q^\pi_{\hat{M}}$, we have

$$\left\| Q^\pi_{\hat{M}_D} - Q^\pi_M \right\|_\infty \leqslant \frac{1}{1-\gamma} \|Q_1 - Q_0\|_\infty ,$$

which completes the proof.

3.6.3 *Proof of Theorem 3.3*

Sketch of the proof
The first steps of the proof are identical to the proof of Theorem 3.2 and we start directly from Equation 3.12. Instead of using the general Hoeffding's inequality and then taking a union bound over all policies, this theorem makes use of the Rademacher complexity (which constrains the set of policies for a given Rademacher complexity) so as to provide a bound in probability which is potentially much tighter.

Let us denote $\varphi, \varphi' \in \phi(\mathcal{H})$ and $f^\pi_{M,\phi}$ as the mapping of $(r, \varphi') \to r + \gamma V^\pi_M(\varphi')$. Equation 3.12 can be rewritten by making use of the following equation for a given φ, a and π:

$$\begin{aligned}
&\left\| \hat{R}(\varphi, a) + \gamma \sum_{\varphi' \in \phi(\mathcal{H})} \hat{T}(\varphi, a, \varphi') V^\pi_M(\varphi') - Q^\pi_M(\varphi, a) \right\|_\infty \\
&= \left\| \frac{1}{n} \sum_{r, \varphi' \in D_{\phi,a}} f^\pi_{M,\phi}(r, \varphi') - \mathop{\mathbb{E}}_{r, \varphi' \sim \mathbb{P}_{\varphi,a}} \left(f^\pi_{M,\phi}(r, \varphi') \right) \right\|_\infty ,
\end{aligned}$$

where

- $r, \varphi' \in D_{\varphi,a}$ means that (r, φ') is a pair of reward r, next state representation φ' corresponding to the pair state representation and action (φ, a) in the dataset D;

- $\mathbb{P}_{\varphi,a}$ is the true distribution of reward r, next history φ' corresponding to the pair history-action (φ, a) (i.e., when $D = D_\infty$).

By noticing that $f^\pi_{M,\phi}$ is bounded in $[0, R_{max}/(1-\gamma)]$ (with $\mathcal{R} \in [0, R_{max}]$), we have the following bound that makes use of the Rademacher complexity (see [Bartlett and Mendelson, 2002]) for each (φ, a) w.p. $\geqslant 1-\delta/(|\phi(\mathcal{H})||A|)$:

$$\forall f^\pi_{M,\phi} \in \mathcal{F}_{M,\phi} : \left\| \frac{1}{n} \sum_{r,\varphi' \in D_{\varphi,a}} f^\pi_{M,\phi}(r,\varphi') - \mathop{\mathbb{E}}_{r,\varphi' \sim \mathbb{P}(\varphi,a)} (f^\pi_{M,\phi}(r,\varphi')) \right\|_\infty$$
$$\leqslant 2\hat{\mathfrak{R}}_{D_{\varphi,a}}(\mathcal{F}_{M,\phi}) + 3\frac{R_{max}}{1-\gamma}\sqrt{\frac{1}{2n} ln(\frac{4|\phi(\mathcal{H})||A|}{\delta})},$$

where $\hat{\mathfrak{R}}_D$ is the empirical Rademacher complexity and theorem 3.3 follows by taking a union bound on all (φ, a).

3.6.4 *Q-learning with neural network as a function approximator: technical details of Figure 3.5*

The neural network is made up of three intermediate fully connected layers with 20, 50 and 20 neurons with ReLu activation function and is trained with Q-learning. Weights are initialized with a Glorot uniform initializer [Glorot and Bengio, 2010]. It is trained using a target Q-network with a freeze interval (see [Mnih et al., 2015]) of 100 mini-batch gradient descent steps. It uses an RMSprop update rule [Tieleman, 2012] (learning rate of 0.005, rho=0.9), mini-batches of size 32 and 20000 mini-batch gradient descent steps.

4

HOW TO DISCOUNT DEEP REINFORCEMENT LEARNING

4.1 INTRODUCTION

In the previous chapter, the sequential decision-making task has been considered in the context of limited data. It was described why it may be favorable to concede a bias so as to reduce overfitting (and thus obtain an overall improved performance). These considerations are also of interest in online settings (i.e., when additional data are gathered while learning the policy). Indeed, in that context, the data obtained at the beginning is severely limited because (i) only a few tuples have been gathered and (ii) the tuples gathered are usually rather uninformative because the agent does not have directly a good exploration/exploitation policy. In this chapter, we investigate the role of a dynamic discount factor in the online setting. In a first phase in the online learning, choosing a low discount factor may help avoid overfitting. When the data gathered become more informative, the ideal approach is then to target a higher discount factor to lower the asymptotic bias of the policy.

In addition to the bias-overfitting tradeoff, the discount factor also plays a key role in the value iteration algorithms. In particular, when bootstrapping is used in conjunction with neural networks, the risk of instabilities (and overestimation of the value function) is stronger when the discount factor is close to one. This effect is due to the mappings used in the value iteration algorithms (Equation 2.9 for the Q-learning algorithm) that propagate errors more strongly with a high discount factor. This effect is discussed in [Gordon, 1999] with the notion of non-expansion/expansion mappings.

In this chapter, we investigate the possibility of reducing both overfitting and instabilities during learning by working on an increasing discount factor. The goal is to reduce errors and instabilities during the deep Q-learning iterations while still targeting a low asymptotic bias. To illustrate our approach, we use the benchmark proposed in [Mnih et al., 2015]. In this context, we empirically show that an increasing discount factor improves the quality of the learning process. We also discuss the role of the learning rate as well as the level of exploration.

Besides the technical motivations from bias-overfitting and propagation of errors, another motivation for this work comes from cognitive science and, in particular, from empirical studies about the attentional and cognitive mechanisms in delay of gratification. One well-known experiment in the domain was a series of studies in which a child was offered a choice between one small reward provided im-

mediately or two small rewards if they waited for a short period ("marshmallow experiment" [Mischel et al., 1972]). The capacity to wait longer for the preferred rewards seems to develop markedly only until about ages 3-4. By 5 years old, most children are able to demonstrate better self-control by gaining control over their immediate desires. According to this theory, it seems plausible that following immediate desires at a young age is a better way to develop children's abilities and that delaying strategies is only advantageous afterwards when pursuing longer term goals. Similarly, reinforcement learning may also have an advantage of starting to learn by maximizing rewards on a short-term horizon and progressively giving more weights to delayed rewards.

4.2 BENCHMARK DESCRIPTION

In this chapter, the task is described by a stochastic discrete-time system whose dynamics can be described by the following equation:

$$H_{t+1} = f(H_t, a_t, w_t), \tag{4.1}$$

where for all t, H_t is an history of observations up to time t, the action a_t is an element of the action space $\mathcal{A}$ and the random disturbance w_t is an element of the disturbance space $\mathcal{W}$ generated by a conditional probability distribution $w_t \sim P(.|H_t)$. In our experiments, the dynamics will be provided by the Atari emulator, the action space is the set of legal game actions $\mathcal{A} = \{1, ..., N_{\mathcal{A}}\}$ and the policies will be built from a mapping $\phi(H_t)$ that is made up of the last four frames of the observed pixels from the screen.

4.3 EXPERIMENTS

The deep Q-learning algorithm described in [Mnih et al., 2015] is used as a benchmark (see Section 2.3.4 for an overview). All hyperparameters are kept identical if not stated differently. The main modification is the discount factor, which is increased at every epoch (250000 steps) with the following formula:

$$\gamma_{k+1} = 1 - 0.98(1 - \gamma_k). \tag{4.2}$$

At every epoch, the learned policies are evaluated for 125000 steps with an ϵ-greedy policy identical to the original benchmark with $\epsilon_{test} = 0.05$. The reported scores are the average episode score (the evaluation episodes were not truncated at 5 minutes as in the reported results in [Mnih et al., 2015]). Each game is simulated 5 times for each configuration with varying seeds, and the results are reported in Figure 4.1. It can be observed that by simply using an increasing discount factor, learning is faster for four out of the five

tested games and similar for the remaining game. We attribute this faster policy improvement to less errors and instability in the learning process. As a side note, another advantage of starting with a low discount factor is that it provides more robustness with respect to the initialization of the neural network weights.

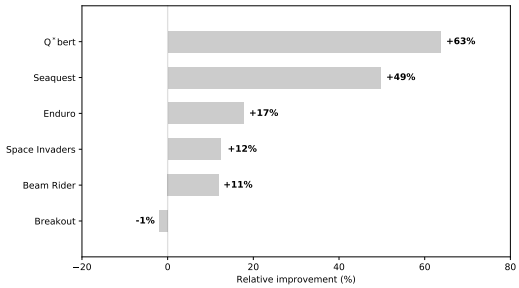


Figure 4.1: Summary of the results for an increasing discount factor. Reported scores are the relative improvements after 20M steps between an increasing discount factor and a constant discount factor set to its final value $\gamma = 0.99$. Details are given in Appendix - Table 4.1. The reported scores are the highest average episode scores during the evaluation step.

4.3.1 *Divergence of DQN*

We are now interested in obtaining more insights on what happens in the learning process at low and high discount factors. We use the experimental rule from Equation 4.2 and either let the discount factor increase or keep it constant when it attains 0.99 (at 20M steps). The results are illustrated on Figure 4.2 for two different games. It is shown that increasing γ without additional care severely degrades the score obtained beyond $\gamma \approx 0.99$. By looking at the average V value, it can be seen that overestimation is particularly severe.

4.3.2 *Further improvement with an adaptive learning rate*

Since overfitting and instability are only severe when the discount factor is high, we now study the possibility to use a more aggressive learning rate in the neural network when working at a low discount factor because potential errors would have less impact at this stage. The learning rate is then reduced along with the increasing discount factor so as to end up with a (more) stable neural Q-learning function. Note that a lower learning rate also allows the neural network to suffer less forgetting of past batches of data, thus it takes into account an artificially larger replay memory, which limits the risk of overfitting (see previous chapter).

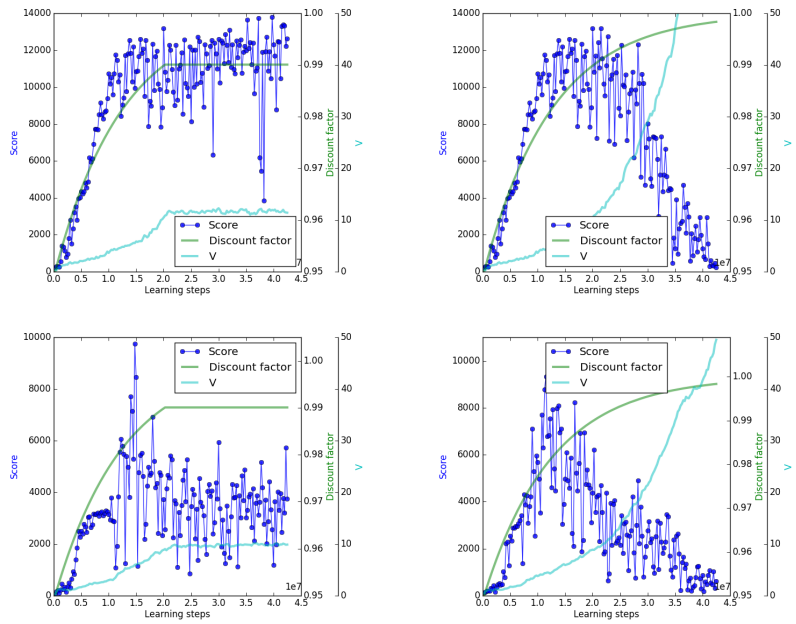


Figure 4.2: Illustration for the game Q*bert (top) and Seaquest (bottom) for a discount factor γ kept at 0.99 after 20M learning steps (two plots on the left), as well as a discount factor that keeps increasing to a value close to 1 (two plots on the right). The average V-value function over the test epochs is provided in light blue.

We start with a learning rate of 0.005, twice as big as the one considered in the original benchmark, and we use the following simple rule at every epoch:

$$\alpha_{k+1} = 0.98\alpha_k.$$

With γ that follows Equation 4.2 and is kept constant when it attains 0.99, we manage to improve further the score obtained on all of the games tested. Results are reported in Figure 4.3, and an illustration is given for two different games in Figure 4.4. It can be noted that the average value function V decreases when γ is held fixed and when the learning rate is lowered. This is a sign of a decrease of the overestimations of the Q-value function.

4.3.3 *Exploration / exploitation dilemma*

One potential side effect of our approach comes from the fact that errors and instabilitites in the policy may also have positive impacts. Indeed they implicitly increase exploration in a way that allows "temporally-extended" exploration similar to what is described in [Osband et al., 2016]. When using a lower discount factor, it reduces errors and instabilitites, and it therefore actually decreases exploration. This opens up the risk of falling into a local optimum in the value iteration learning. This was particularly striking for the game Seaquest

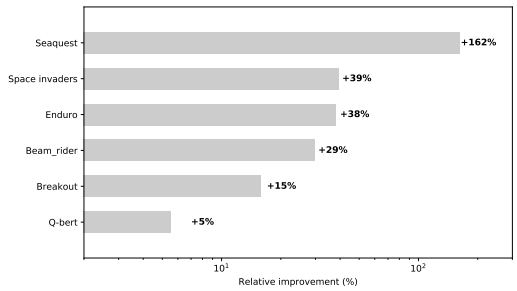


Figure 4.3: Summary of the results for a decreasing learning rate. Reported scores are the improvement after 50M steps when using both a dynamic discount factor and a dynamic learning rate as compared to only a dynamic discount factor. Details are given in Appendix - Table 4.2.

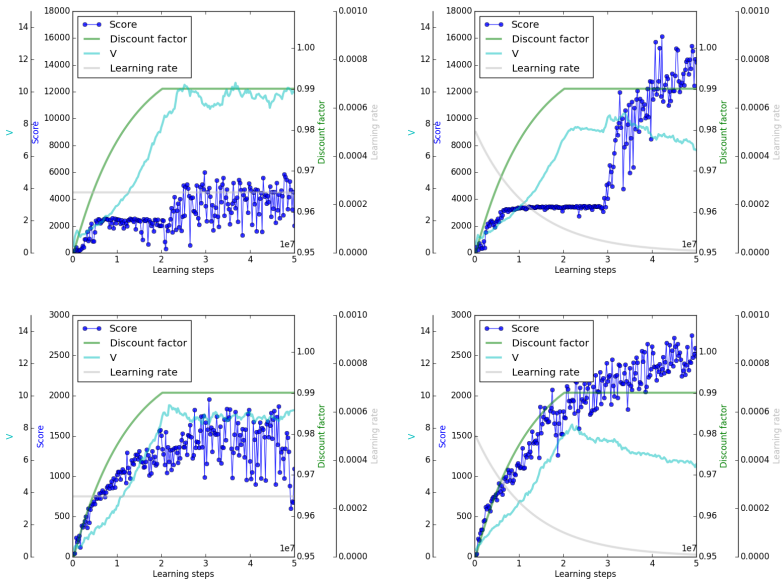


Figure 4.4: Illustration for the game Seaquest (top) and space invaders (bottom). On the left is the deep Q-network with original learning rate ($\alpha = 0.00025$) and on the right is the setting with a decreasing learning rate.

where the agent gets a score lower than the optimal while being unable to discover some parts of the environment. An algorithm that increases the level of exploration may overcome this problem. In order to illustrate this, a simple rule has been applied in the case of the game Seaquest as can be seen on Figure 4.5. This rule adapts the exploration during the training process in the ϵ-greedy action selection until the agent was able to get out of the local optimum (the same behavior has been obtained with different seeds).

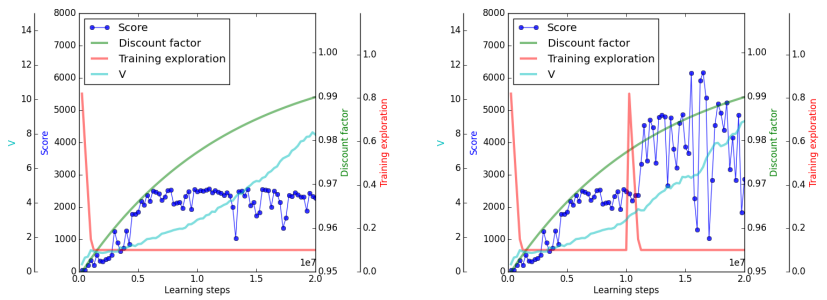


Figure 4.5: Illustration for the game Seaquest. On the left, the flat exploration rate fails to get the agent out of a local optimum. On the right, a simple rule that increases exploration allows the agent to get out of the local optimum.

4.3.4 *Towards an adaptive algorithm*

Building on the ideas discussed in the previous chapter and in this chapter, Figure 4.6 represents a general update scheme that could further improve the performance of deep reinforcement learning algorithms (with the value-based approach in this figure). In particular, in the hypothesis that the main difficulty is to gather a lot of tuples, a robust algorithm should be able to adapt the discount factor along with the learning rate and possibly the level of exploration (or even e.g., the neural network architecture). In general, an adaptive algorithm could compare several set of hyperparameters in parallel and the hyperparameters could move towards the set of hyperparameters of the best performing one(s). For instance, one can expect that, following the results described above, the discount factor would automatically be rather low at the beginning of learning and would then increase when more informative data becomes available.

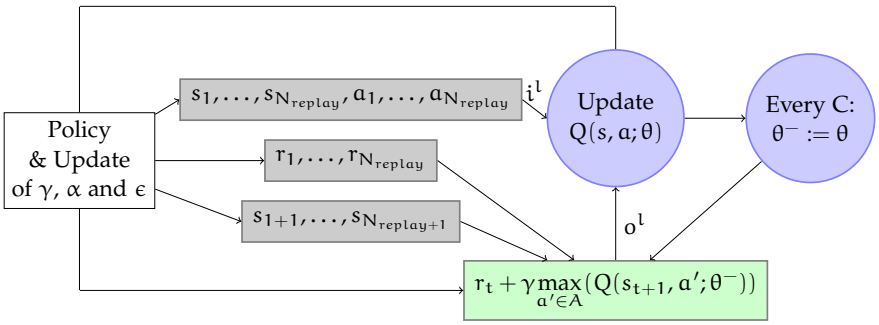


Figure 4.6: Sketch of the adaptive algorithm for deep reinforcement learning that dynamically modifies the discount rate γ, the learning rate α as well as the level of exploration ϵ.

4.4 CONCLUSION

This chapter introduced an approach to speed up the convergence and improve the quality of the learned Q-function in deep reinforcement learning algorithms. It works by adapting the discount factor and possibly the learning rate along the way to convergence. We used the deep Q-learning algorithms for Atari 2600 computer games as a benchmark, and our approach showed improved performances for the 6 tested games. As future work, it would be of interest to combine this approach with the recent advances in deep reinforcement learning (see Chapter 2 for an overview of methodological improvements to DQN and alternatives).

4.5 APPENDIX

	Seaquest	Q*bert	Breakout	Enduro	Beam Rider	Space Invaders
$\gamma = 0.99$,	4766	7930	346	817	8379	2023
20M steps	4774	8262	359	950	9013	1071
	2941	7527	372	821	9201	1473
	3597	8214	374	863	8496	1844
	4280	7867	365	777	8408	1352
Average	**4072**	**7960**	**363**	**846**	**8699**	**1553**
Increasing γ,	2570	13073	351	929	9011	1807
20M steps	2575	12873	361	1031	10160	1759
	11717	13351	362	1099	9880	1723
	11030	12828	354	978	9263	1761
	2583	13063	351	945	10389	1671
Average	**6095**	**13031**	**356**	**996**	**9741**	**1744**

Table 4.1: Summary of the results for an increasing discount factor.

	Seaquest	Q*bert	Breakout	Enduro	Beam Rider	Space Invaders
Fixed γ and α, 50M steps	6150	12958	401	817	9606	1976
Varying γ, fixed α, 50M steps	6016	13997	389	929	10677	1954
Variable γ and α, 50M steps	16124	14996	423	1129	12473	2750

Table 4.2: Summary of the results for an increasing discount factor associated with a decreasing learning rate.

Part II

MICROGRID OPERATION

5

TOWARDS THE MINIMIZATION OF THE LEVELIZED ENERGY COSTS OF MICROGRIDS USING BOTH LONG-TERM AND SHORT-TERM STORAGE DEVICES

5.1 INTRODUCTION

Economies of scale of conventional power plants have progressively led to the development of the very large and complex electrical networks that we know today. These networks transmit and distribute the power generated by these power plants to the consumers. Over recent years, a new trend opposing this centralization of power facilities has been observed, resulting from the drop in price of distributed generation, mainly solar photovoltaic (PV) panels [Bazilian et al., 2013]. Due to this effect, it is expected that in the future, small scale industries and residential consumers of electricity will rely more and more on local renewable energy production capacities for covering, at least partially, their need for electrical power. This leads to the creation of the so-called microgrids that are electrical systems which include loads and distributed energy resources that can be operated in parallel with the broader utility grid or as an electrical island. State-of-the-art issues and feasible solutions associated with the deployment of microgrids are discussed in [Ustun et al., 2011].

Due to the fluctuating nature of renewable energy sources (RES) (mainly solar and wind energy), small businesses and residential consumers of electricity may also be tempted to combine their local power plants with storage capacities. In principle, this would, at least partially, allow themselves freedom from using the network, enabling balancing their own electricity generation with their own consumption. This would result in paying less in transmission and distribution fees which typically make up around 50% of their electricity bill. Many different technologies are available for energy storage as discussed in the literature (e.g., [Ferreira et al., 2013]). On the one hand, hydrogen is often considered as a storage solution to be combined with RES [Krajačić et al., 2009; Connolly et al., 2011], mainly due to its high capacity potential that makes it suitable for long-term storage [François-Lavet et al., 2014; Armaroli and Balzani, 2011]. On the other hand, batteries are often used to ensure sufficient peak power storage and peak power generation [Schoenung, 2001]. In this chapter we focus on the specific case of microgrids that are powered by PV panels combined with both hydrogen-based storage technologies (electrolysis combined with fuel cells) and batteries (such as, for instance, $LiFePO_4$ batteries). These two types of storage aim at fulfilling, at best, the demand by addressing the seasonal and

daily fluctuations of solar irradiance. Distinguishing short- from long-term storage is mainly a question of cost: batteries are currently too expensive to be used for addressing seasonal variations.

One of the main problems to be addressed in the field of microgrids is how to perform optimal sizing. The main challenge when sizing microgrids comes from the need to determine and simulate their operation (the dispatch strategy) using historical data of the loads and of the RES.

In this chapter, we first propose a novel and detailed formalization of the problem of sizing and operating microgrids under different assumptions on the components used (PV panels and storage systems). In that context, we show how to optimally operate a microgrid so that it minimizes a levelized energy cost (LEC) criterion in the context where the energy production and demand are known. We show that this optimization step can be achieved efficiently using linear programming techniques (thanks to the assumptions on the components used and with the help of auxiliary variables in the linear program). We then show that this optimization step can also be used to address the problem of optimal sizing of the microgrid (still with the hypothesis that production and demand are known), for which we propose a (potentially) robust approach by considering several energy production and consumption scenarios. We run experiments using real data corresponding to the case of typical residential consumers of electricity located in Spain and in Belgium. Note that this chapter focuses on the production planning and optimal sizing of the microgrid without uncertainty. The real-time control under uncertainty of microgrids is studied in the next chapter.

Subsequently, the chapter is organized as follows. A detailed formalization of the microgrid framework is proposed in Section 5.2 and several optimization schemes for minimizing the LEC are introduced in Section 5.3. The simulation results for Belgium and Spain are finally reported in Section 5.4 while Section 5.5 provides the conclusion.

5.2 FORMALIZATION AND PROBLEM STATEMENT

In this section we provide a generic model of a microgrid powered by PV panels combined with batteries and hydrogen-based storage technologies. We formalize its constitutive elements as well as its dynamics within the surrounding environment. For the sake of clarity, we first define the space of exogenous variables and then gradually build the state and action spaces of the system. The components of these two latter spaces will be related to either the notion of *infrastructure* or the notion of *operation* of the microgrid. We then characterize the problem of sizing and control that we want to address using an optimality criterion, which leads to the formalization of an optimal sequential decision-making problem. The

evolution of the system is described as a discrete-time process over a finite time-horizon. We denote by $\mathcal{T}$ the set $\{1, ..., H\}$ of time periods and by Δt the duration of one time step. We use subscript t to reference exogenous variables, state and actions at time step t. Finally we introduce the notion of LEC and discuss how it can be used as an optimality criterion.

5.2.1 *Exogenous variables*

We start with a definition of the microgrid's environment by defining the space of exogenous variables that affect the microgrid but on which the latter has no control. Assuming that there exists, respectively, J, L, and M different photovoltaic, battery, and hydrogen storage technologies, and denoting the environment space by $\mathcal{E}$, we can define the time-varying environment vector $\mathbf{E}_t$ as:

$$\begin{aligned}\mathbf{E}_t = (c_t, i_t, \mathbf{e}^{PV}_{1,t}, ..., \mathbf{e}^{PV}_{J,t}, \mathbf{e}^{B}_{1,t}, ..., \mathbf{e}^{B}_{L,t}, \mathbf{e}^{H_2}_{1,t}, ..., \mathbf{e}^{H_2}_{M,t}, \boldsymbol{\mu}_t) \in \mathcal{E},\ \forall t \in \mathcal{T} \\ \text{and with } \mathcal{E} = \mathbb{R}^{+2} \times \prod_{j=1}^{J} \mathcal{E}^{PV}_j \times \prod_{l=1}^{L} \mathcal{E}^{B}_l \times \prod_{m=1}^{M} \mathcal{E}^{H_2}_m \times \mathcal{I},\end{aligned} \tag{5.1}$$

where:

- c_t [W] $\in \mathbb{R}^+$ is the electricity demand within the microgrid;
- i_t [W/m^2 or W/W$_p$] $\in \mathbb{R}^+$ denotes the solar irradiance incident to the PV panels;
- $\mathbf{e}^{PV}_{j,t} \in \mathcal{E}^{PV}_j$, $\forall j \in \{1, ..., J\}$, models a photovoltaic technology in terms of cost $c^{PV}_{j,t}$ [€/m^2], lifetime $L^{PV}_{j,t}$ [s] and efficiency $\eta^{PV}_{j,t}$ to convert solar irradiance to electrical power:

$$\begin{aligned}\mathbf{e}^{PV}_{j,t} = (c^{PV}_{j,t}, L^{PV}_{j,t}, \eta^{PV}_{j,t}) \in \mathcal{E}^{PV}_j,\ \forall j \in \{1, ..., J\} \\ \text{and with } \mathcal{E}^{PV}_j = \mathbb{R}^{+2} \times]0, 1]\,;\end{aligned} \tag{5.2}$$

- $\mathbf{e}^{B}_{l,t} \in \mathcal{E}^{B}_l$, $\forall l \in \{1, ..., L\}$, represents a battery technology in terms of cost $c^{B}_{l,t}$ [€/Wh], lifetime L^{B}_l [s], cycle durability $D^{B}_{l,t}$, power limit for charge and discharge $P^{B}_{l,t}$ [W], charge efficiency $\eta^{B}_{l,t}$, discharge efficiency $\zeta^{B}_{l,t}$, and charge retention rate $r^{B}_{l,t}$ [s^{-1}]:

$$\begin{aligned}\mathbf{e}^{B}_{l,t} = (c^{B}_{l,t}, L^{B}_{l,t}, P^{B}_{l,t}, \eta^{B}_{l,t}, \zeta^{B}_{l,t}, r^{B}_{l,t}) \in \mathcal{E}^{B}_l,\ \forall l \in \{1, ..., L\} \\ \text{and with } \mathcal{E}^{B}_l = \mathbb{R}^{+3} \times]0, 1]^3\,;\end{aligned} \tag{5.3}$$

- $\mathbf{e}^{H_2}_{m,t} \in \mathcal{E}^{H_2}_m$, $\forall m \in \{1, ..., M\}$, denotes a hydrogen storage technology in terms of cost $c^{H_2}_{m,t}$ [€/W$_p$], lifetime $L^{H_2}_{m,t}$ [s], maximum capacity $R^{H_2}_{m,t}$ [Wh], electrolysis efficiency $\eta^{H_2}_{m,t}$ (i.e., when

storing energy), fuel cells efficiency $\zeta_{m,t}^{H_2}$ (i.e., when delivering energy), and charge retention rate $r_{m,t}^{H_2}$ [s^{-1}]:

$$\mathbf{e}_{m,t}^{H_2} = (c_{m,t}^{H_2}, L_{m,t}^{H_2}, R_{m,t}^{H_2}, \eta_{m,t}^{H_2}, \zeta_{m,t}^{H_2}, r_{m,t}^{H_2}) \in \mathcal{E}_m^{H_2},\ \forall m \in \{1, ..., M\} \text{ and with } \mathcal{E}_m^{H_2} = \mathbb{R}^{+3} \times]0, 1]^3 . \tag{5.4}$$

Finally, the components denoted by $\boldsymbol{\mu}_t \in \mathcal{I}$ represent the model of interaction. By model of interaction we mean all the information that is required to manage and evaluate the costs (or benefits) related to electricity exchanges between the microgrid and the rest of the system. We assume that $\boldsymbol{\mu}_t$ is composed of two components k and β:

$$\boldsymbol{\mu}_t = (k, \beta) \in \mathcal{I},\ \forall\ t\ \in\ \mathcal{T} \text{ and with } \mathcal{I} = \mathbb{R}^{+2} . \tag{5.5}$$

The variable β characterizes the price per kWh at which it is possible to sell energy to the grid (it is set to 0 in the case of an off-grid microgrid). The variable k refers to the cost incurred per kWh that is not supplied within the microgrid. In a connected microgrid, k corresponds to the price at which electricity can be bought from outside the microgrid. In the case of an off-grid microgrid, the variable k characterizes the penalty associated with a failure of the microgrid to fulfill the demand. This penalty is known as the value of loss load and corresponds to the amount that consumers of electricity would be willing to pay to avoid a disruption to their electricity supply.

5.2.2 *State space*

Let $\mathbf{s}_t \in \mathcal{S}$ denote a time varying vector characterizing the microgrid's state at time $t \in \mathcal{T}$:

$$\mathbf{s}_t = (\mathbf{s}_t^{(s)}, \mathbf{s}_t^{(o)}) \in \mathcal{S},\ \forall t \in \mathcal{T} \text{ and with } \mathcal{S} = \mathcal{S}^{(s)} \times \mathcal{S}^{(o)} , \tag{5.6}$$

where $\mathbf{s}_t^{(s)} \in \mathcal{S}^{(s)}$ and $\mathbf{s}_t^{(o)} \in \mathcal{S}^{(o)}$ respectively represent the state information related to the infrastructure and to the operation of the microgrid.

5.2.2.1 *Infrastructure state*

The infrastructure state vector $\mathbf{s}_t^{(s)} \in \mathcal{S}^{(s)}$ gathers all the information about the physical and electrical properties of the devices that constitute the microgrid. Its components can only change because of investment decisions or due to aging of the devices. In particular, we define this vector as:

$$\mathbf{s}_t^{(s)} = (x_t^{PV}, x_t^B, x_t^{H_2}, L_t^{PV}, L_t^B, L_t^{H_2}, P_t^B, R_t^{H_2}, \eta_t^{PV}, \eta_t^B, \eta_t^{H_2}, \zeta_t^B, \zeta_t^{H_2}, r_t^B, r_t^{H_2}) \in \mathcal{S}^{(s)} ,\forall t \in \mathcal{T} \text{ and with } \mathcal{S}^{(s)} = \mathbb{R}^{+8} \times]0, 1]^7 ,$$

(5.7)

where x_t^{PV} [m^2], x_t^B [Wh], and $x_t^{H_2}$ [W_p] denote, respectively, the sizing of the PV panels, battery and hydrogen storage. The other components have the same meaning than the exogenous variables using a similar symbol, with the difference here that they are specific to the devices that are present at time $t \in \mathcal{T}$ in the microgrid. Note that by using such a representation, we consider that, for each device type, a single device can operate in the microgrid. In other words, an investment decision for a device type substitutes any prior investment.

5.2.2.2 *Operation state*

For the devices with storage capacities (i.e., battery and hydrogen storage) the information provided by the environment vector $\mathbf{E}_t$ and by the infrastructure state vector $\mathbf{s}_t^{(s)}$ is not sufficient to determine the set of their feasible power injections or demands. Additional information that corresponds to the amount of energy stored in these devices for each time period is required. For this reason we introduce the operation state vector $\mathbf{s}_t^{(o)}$:

$$\mathbf{s}_t^{(o)} = (s_t^B, s_t^{H_2}) \in \mathcal{S}^{(o)},\ \forall t \in \mathcal{T} \text{ and with } \mathcal{S}^{(o)} = \mathbb{R}^{+2}\,, \tag{5.8}$$

where s_t^B [Wh] is the level of charge of the battery and with $s_t^{H_2}$ [Wh] the level of charge of the hydrogen storage.

5.2.3 *Action space*

As for the state space, each component of the action vector $\mathbf{a}_t \in \mathcal{A}$ can be related to either sizing or control, the former affecting the infrastructure of the microgrid while the latter affects its operation. We define the action vector as:

$$\mathbf{a}_t = (\mathbf{a}_t^{(s)}, \mathbf{a}_t^{(o)}) \in \mathcal{A}_t,\ \forall t \in \mathcal{T} \text{ and with } \mathcal{A} = \mathcal{A}^{(s)} \times \mathcal{A}_t^{(o)}\,, \tag{5.9}$$

where $\mathbf{a}_t^{(s)} \in \mathcal{A}^{(s)}$ relates to sizing actions and $\mathbf{a}_t^{(o)} \in \mathcal{A}_t^{(o)}$ to control actions.

5.2.3.1 *Sizing actions*

The sizing actions correspond to investment decisions. For each device type, it defines the sizing of the device to install in the microgrid and its technology:

$$\mathbf{a}_t^{(s)} = (a_t^{PV}, a_t^B, a_t^{H_2}, j_t, l_t, m_t) \in \mathcal{A}^{(s)},\ \forall t \in \mathcal{T} \tag{5.10}$$

$$\text{and with } \mathcal{A}^{(s)} = \mathbb{R}^{+3} \times \{1, ..., J\} \times \{1, ..., L\} \times \{1, ..., M\}\,, \tag{5.11}$$

where a_t^{PV} [m^2], a_t^B [Wh], and $a_t^{H_2}$ [Wp] denote, respectively, the new sizing at time $t+1 \in \mathcal{T}$ of the PV panels, battery and hydrogen storage. Discrete variables j_t, l_t, and m_t correspond to indices that indicate the selected technology from the environment vector for PV panels, battery, and hydrogen storage, respectively. When a sizing variable (i.e., a_t^{PV}, a_t^B, or $a_t^{H_2}$) is equal to zero, it means that there is no new installation for the corresponding device type and that the present device, if it exists, remains in operation.

5.2.3.2 *Operational planning*

A microgrid featuring PV, battery and storage using H_2 has two control variables that correspond to the power exchanges between the battery, the hydrogen storage, and the rest of the system:

$$\mathbf{a}_t^{(o)} = (p_t^B, p_t^{H_2}) \in \mathcal{A}_t^{(o)}, \forall t \in \mathcal{T}, \tag{5.12}$$

where p_t^B [W] is the power provided to the battery and where $p_t^{H_2}$ [W] is the power provided to the hydrogen storage device. These variables are positive when the power flows from the system to the devices and negative if it flows in the other direction. Note that the set $\mathcal{A}_t^{(o)}$ of control actions is time dependent. This comes from the fact that the feasible power exchanges with these devices depend on their capacity and level of charge. We have, $\forall t \in \mathcal{T}$:

$$\begin{aligned}\mathcal{A}_t^{(o)} = &\left([-\zeta_t^B s_t^B, \frac{x_t^B - s_t^B}{\eta_t^B}] \cap [-P_t^B, P_t^B] \right) \\ &\times \left([-\zeta_t^{H_2} s_t^{H_2}, \frac{R_t^{H_2} - s_t^{H_2}}{\eta_t^{H_2}}] \cap [-x_t^{H_2}, x_t^{H_2}] \right) ,\end{aligned} \tag{5.13}$$

which expresses 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.

5.2.4 *Dynamics*

Using the formalism proposed above, the dynamics of the microgrid follows the following discrete-time equation:

$$\mathbf{s}_{t+1} = f(\mathbf{s}_t, \mathbf{a}_t), \forall t \in \mathcal{T} \text{ and with } (\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}) \in \mathcal{S} \times \mathcal{A}_t \times \mathcal{S} . \tag{5.14}$$

The dynamics specific to the infrastructure state $\mathbf{s}_t^{(s)} \in \mathcal{S}^{(s)}$ are straightforward and can be written, $\forall t \in \mathcal{T}$:

$$\begin{aligned}&(x_{t+1}^{PV}, L_{t+1}^{PV}, \eta_{t+1}^{PV}) \\ &\quad = \begin{cases} (a_t^{PV}, L_{j_t,t}^{PV}, \eta_{j_t,t}^{PV}) & \text{if } a_t^{PV} > 0, \\ (0, 0, \eta_t^{PV}) & \text{if } L_t^{PV} \leqslant 1, \\ (x_t^{PV}, L_t^{PV} - 1, \eta_t^{PV}) & \text{otherwise,} \end{cases}\end{aligned} \tag{5.15}$$

$$(x^B_{t+1}, L^B_{t+1}, P^B_{t+1}, \eta^B_{t+1}, \zeta^B_{t+1}, r^B_{t+1}) = \begin{cases} (a^B_t, L^B_{l_t,t}, P^B_{l_t,t}, \eta^B_{l_t,t}, \zeta^B_{l_t,t}, r^B_{l_t,t}) & \text{if } a^B_t > 0, \\ (0, 0, 0, \eta^B_t, \zeta^B_t, r^B_t) & \text{if } L^B_t \leqslant 1, \\ (x^B_t, L^B_t - 1, P^B_t, \eta^B_t, \zeta^B_t, r^B_t) & \text{otherwise,} \end{cases} \tag{5.16}$$

$$(x^{H_2}_{t+1}, L^{H_2}_{t+1}, R^{H_2}_{t+1}, \eta^{H_2}_{t+1}, \zeta^{H_2}_{t+1}, r^{H_2}_{t+1}) = \begin{cases} (a^{H_2}_t, L^{H_2}_{m_t,t}, R^{H_2}_{m_t,t}, \eta^{H_2}_{m_t,t}, \zeta^{H_2}_{m_t,t}, r^{H_2}_{m_t,t}) & \text{if } a^{H_2}_t > 0, \\ (0, 0, 0, \eta^{H_2}_t, \zeta^{H_2}_t, r^{H_2}_t) & \text{if } L^{H_2}_t \leqslant 1, \\ (x^{H_2}_t, L^{H_2}_t - 1, R^{H_2}_t, \eta^{H_2}_t, \zeta^{H_2}_t, r^{H_2}_t) & \text{otherwise,} \end{cases} \tag{5.17}$$

which describes that a device is either replaced because of a new investment or because of aging. At the end of the device's lifetime, it is discarded from the microgrid. Note that a more advanced model could include aging rules for the other physical properties of the devices (i.e., efficiency, energy retention, capacity and power limit) but this is outside the scope of the present work.

Concerning the dynamics of the operation state $\mathbf{s}^{(o)}_t \in \mathcal{S}^{(o)}$, we have to ensure that the charge level of a storage device is reset to zero when it is replaced by a new investment. In addition, the correct efficiency factor differs depending on the direction of the power flow:

$$s^B_{t+1} = \begin{cases} 0 & \text{if } a^B_t > 0, \\ r^B_t s^B_t + \eta^B_t p^B_t \Delta t & \text{if } p^B_t \geqslant 0, \\ r^B_t s^B_t + \frac{p^B_t \Delta t}{\zeta^B_t} & \text{otherwise,} \end{cases} \tag{5.18}$$

$$s^{H_2}_{t+1} = \begin{cases} 0 & \text{if } a^{H_2}_t > 0, \\ r^{H_2}_t s^{H_2}_t + \eta^{H_2}_t p^{H_2}_t \Delta t & \text{if } p^{H_2}_t \geqslant 0, \\ r^{H_2}_t s^{H_2}_t + \frac{p^{H_2}_t \Delta t}{\zeta^{H_2}_t} & \text{otherwise.} \end{cases} \tag{5.19}$$

5.2.5 *Problem statement formalization*

We now rely on the introduced formalism to define three optimization problems of increasing complexity. The first one focuses on the optimal operation of a microgrid, while the two others respectively include the optimal and robust sizing of the microgrid.

5.2.5.1 *Optimal operation*

Let $\mathcal{G}_T$ be the set of all positive scalar functions defined over the set of ordered lists of T triplets (state, action, environment) :

$$\mathcal{G}_T = \left\{ G_T : (\mathcal{S} \times \mathcal{A}_t \times \mathcal{E})^T \to \mathbb{R}^+ \right\}. \tag{5.20}$$

Problem 1 *Given a function* $G_T \in \mathcal{G}_T$ *and a trajectory* $(E_1, \dots, E_T)$ *of* T *environment vectors, we formalize the problem of optimal operation of a microgrid in the following way:*

$$\begin{aligned} \min_{\substack{a_t \in \mathcal{A}_t, \forall t \in \mathcal{T} \\ s_t \in \mathcal{S}, \forall t \in \mathcal{T} \setminus \{1\}}} \quad & G_T((s_1, a_1, E_1), \dots, (s_T, a_T, E_T)) \\ \text{s.t.} \quad & \mathbf{s}_t = f(\mathbf{s}_{t-1}, \mathbf{a}_{t-1}), \quad \forall t \in \mathcal{T} \setminus \{1\}, \\ & (a_t^{PV}, a_t^{B}, a_t^{H_2}) = (0, 0, 0), \quad \forall t \in \mathcal{T}. \end{aligned}$$

This problem determines the sequence of control variables that leads to the minimization of G_T when the sizing decisions are made once for all at a prior stage $t = 0$. The initial state $\mathbf{s}_1$ of the system contains the sizing information of the microgrid and stands as a parameter of this problem.

5.2.5.2 *Optimal sizing under optimal operation*

Let $\mathcal{G}_0$ be the set of all positive scalar functions defined over the set of (action, environment, T-long environment trajectory) triplets:

$$\mathcal{G}_0 = \left\{ G_0 : (\mathcal{A}_t \times \mathcal{E} \times \mathcal{E}^T) \to \mathbb{R}^+ \right\}. \tag{5.21}$$

Problem 2 *Given a function* $G_0 \in \mathcal{G}_0$*, a function* $G_T \in \mathcal{G}_T$*, an initial environment* E_0 *that describes the available technologies at the sizing step, and a trajectory* $(E_1, \dots, E_T)$ *of* T *environment vectors (compatible with* E_0 *and* a_0*), we formalize the problem of optimal sizing of a microgrid under optimal operation in the following way:*

$$\begin{aligned} \min_{\substack{a_t \in \mathcal{A}_t, s_t \in \mathcal{S}, \\ \forall t \in \{0\} \cup \mathcal{T}}} \quad & G_0(a_0, E_0, E_1, \dots, E_T) + \\ & \quad G_T((s_1, a_1, E_1), \dots, (s_T, a_T, E_T)) \\ \text{s.t.} \quad & \mathbf{s}_t = f(\mathbf{s}_{t-1}, \mathbf{a}_{t-1}), \quad \forall t \in \mathcal{T}, \\ & \mathbf{s}_0 = \mathbf{0}, \\ & (a_t^{PV}, a_t^{B}, a_t^{H_2}) = (0, 0, 0), \quad \forall t \in \mathcal{T}, \end{aligned}$$

with $\mathbf{s}_0$ *being the null vector to model that we start from an empty microgrid.*

This problem determines an initial sizing decision $\mathbf{a}_0$ such that, together with the sequence of control variables over $\{1, \dots, T\}$, it leads to the minimization of $G_0 + G_T$.

5.2.5.3 *Robust sizing under optimal operation*

Let $\mathbf{E}$ be a set of environment trajectories:

$$\mathbf{E} = \{(E_t^1)_{t=1\dots T}, \dots, (E_t^N)_{t=1\dots T}\}, \quad \text{with } E_t^i \in \mathcal{E},\ \forall (t,i) \in \mathcal{T} \times \{1,\dots,N\}. \tag{5.22}$$

Problem 3 *Given a function $G_0 \in \mathcal{G}_0$, a function $G_T \in \mathcal{G}_T$, an initial environment E_0, and a set $\mathbf{E}$ of trajectories of T environment vectors that describes the potential scenarios of operation that the microgrid could face, we formalize the problem of robust sizing of a microgrid under optimal operation in the following way:*

$$\begin{aligned}
\min_{a_0 \in \mathcal{A}_0} \max_{i \in \{1,\dots,N\}} \min_{\substack{a_{i,t} \in \mathcal{A}_{i,t},\\ s_{i,t} \in \mathcal{S},\\ \forall t \in \mathcal{T}}} \quad & G_0(a_0, E_0, E_1^i, \dots, E_T^i) + \\
& G_T((s_{i,1}, a_{i,1}, E_1^i), \dots, (s_{i,T}, a_{i,T}, E_T^i)) \\
\text{s.t.} \quad & \mathbf{s}_{i,t} = f(\mathbf{s}_{i,t-1}, \mathbf{a}_{i,t-1}), \quad \forall t \in \mathcal{T}\backslash\{1\}, \\
& \mathbf{s}_{i,1} = f(\mathbf{s}_0, \mathbf{a}_0), \\
& \mathbf{s}_0 = \mathbf{0}, \\
& (a_{i,t}^{PV}, a_{i,t}^{B}, a_{i,t}^{H_2}) = (0,0,0), \quad \forall t \in \mathcal{T}.
\end{aligned}$$

This robust optimization considers a microgrid under optimal operation and determines the sizing so that, in the worst case scenario, it minimizes the objective function. The innermost min is for the optimal operation, the max is for the worst environment trajectory and the outermost min is the minimization over the investment decisions. The outermost min-max succession is classic in robust optimizations (e.g., [Ben-Tal and Nemirovski, 2002]).

5.2.6 *The specific case of the Levelized Energy Cost*

In this section, we introduce the ρ-discounted levelized energy cost (LEC), denoted LEC_ρ, 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). We then show how to choose functions $G_0 \in \mathcal{G}_0$ and $G_T \in \mathcal{G}_T$ such that Problems 1, 2, and 3 result in the optimization of this economic assessment. Focusing on the decision processes that consist only with an initial investment (i.e., a single sizing decision taking place at $t = 0$) for the microgrid, followed by the control of its operation, we can write the expression for LEC_ρ as

$$LEC_\rho = \frac{I_0 + \sum_{y=1}^{n} \frac{M_y}{(1+\rho)^y}}{\sum_{y=1}^{n} \frac{E_y}{(1+\rho)^y}}, \tag{5.23}$$

where

- n denotes the lifetime of the system in years;
- I_0 corresponds to the initial investment expenditures;
- M_y represents the operational expenses in the year y;
- ϵ_y is electricity consumption in the year y;
- ρ denotes the discount rate which may refer to the interest rate or to the discounted cash flow.

Note that, in the more common context of an electrical generation facility, the LEC_ρ can be interpreted as the price at which the electricity generated must be sold to break even over the lifetime of the project. For this reason, it is often used to compare the costs of different electrical generation technologies. When applied to the microgrid case, it can also be interpreted as the retail price at which the electricity from the grid must be bought in order to face the same costs when supplying a sequence $(\epsilon_1, \dots, \epsilon_n)$ of yearly consumptions.

The initial investment expenditures I_0 and the yearly consumptions ϵ_y are simple to express as a function of the initial sizing decision $\mathbf{a}_0$ and environment vector $\mathbf{E}_0$ for the former, and of the environment trajectory $(E_1, \dots, E_T)$ for the latter. Let $\tau_y \subset \mathcal{T}$ denotes, $\forall y \in \{1, \dots, n\}$, the set of time steps t belonging to year y, we have:

$$I_0 = a_0^{PV} c_0^{PV} + a_0^{B} c_0^{B} + a_0^{H_2} c_0^{H_2} \tag{5.24}$$

$$\epsilon_y = \sum_{t \in \tau_y} c_t \, \Delta t, \, \forall y \in \{1, \dots, n\}. \tag{5.25}$$

From these two quantities, we can define the function $G_0 \in \mathcal{G}_0$ that implements the LEC case as:

$$\begin{aligned} G_0(a_0, E_0, E_1, \dots, E_T) &= \frac{I_0}{\sum_{y=1}^{n} \frac{\epsilon_y}{(1+\rho)^y}} \\ &= \frac{a_0^{PV} c_0^{PV} + a_0^{B} c_0^{B} + a_0^{H_2} c_0^{H_2}}{\sum_{y=1}^{n} \frac{\sum_{t \in \tau_y} c_t \Delta t}{(1+\rho)^y}}, \end{aligned} \tag{5.26}$$

while the remaining term of LEC_ρ defines $G_T \in \mathcal{G}_T$:

$$G_T((s_1, a_1, E_1), \dots, (s_T, a_T, E_T)) = \frac{\sum_{y=1}^{n} \frac{M_y}{(1+\rho)^y}}{\sum_{y=1}^{n} \frac{\epsilon_y}{(1+\rho)^y}}. \tag{5.27}$$

The last quantities to specify are the yearly operational expenses M_y, which correspond to the opposite of the sum over the year $y \in \mathcal{Y}$ of the revenues r_t observed at each time step $t \in \tau_y$ when operating the microgrid:

$$M_y = -\sum_{t \in \tau_y} r_t \,. \tag{5.28}$$

These revenues are more complex to determine than the investment expenditures and depend, among other elements, on the model of interaction μ_t at the time of the operation.

5.2.6.1 *Operational revenues*

The instantaneous operational revenues r_t at time step $t \in \mathcal{T}$ correspond to the reward function of the system. This is a function of the electricity demand c_t, of the solar irradiance i_t, of the model of interaction $\mu_t = (k, \beta)$, and of the control actions $\mathbf{a}_t^{(o)}$:

$$r_t : (c_t, i_t, \mu_t, \mathbf{a}_t^{(o)}) \to \mathbb{R}\,.$$

We now introduce three quantities that are prerequisites to the definition of the reward function:

- ψ_t [kW] $\in \mathbb{R}^+$ is the electricity generated locally by the photovoltaic installation, we have:

$$\psi_t = \eta_t^{PV} x_t^{PV} i_t\,; \tag{5.29}$$

- d_t [kW] $\in \mathbb{R}$ denotes the net electricity demand, which is the difference between the local consumption and the local production of electricity:

$$d_t = c_t - \psi_t\,; \tag{5.30}$$

- δ_t [kW] $\in \mathbb{R}$ represents the power balance within the microgrid, taking into account the contributions of the demand and of the storage devices:

$$\delta_t = -p_t^B - p_t^{H_2} - d_t\,. \tag{5.31}$$

These quantities are illustrated in a diagram of the system in Figure 5.1, which allows for a more intuitive understanding of the power flows within the microgrid.

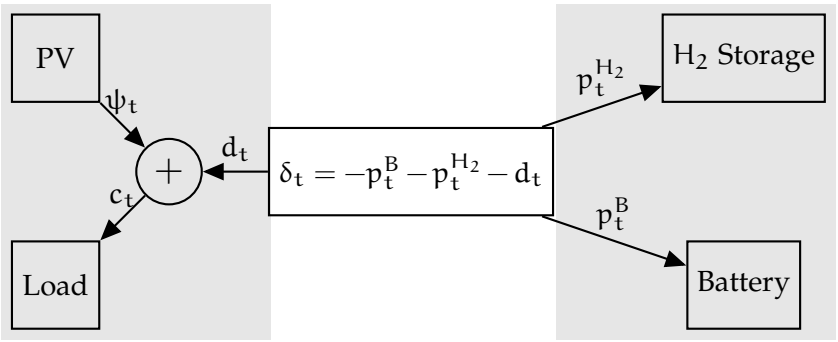


Figure 5.1: Schema of the microgrid featuring PV panels associated with a battery and a hydrogen storage device.

At each time step $t \in \mathcal{T}$, a positive power balance δ_t reflects a surplus of production within the microgrid, while it is negative when the power demand is not met. As the law of conservation of energy requires that the net power within the microgrid must be null, compensation measures are required when δ_t differs from zero. In the

case of a connected microgrid, this corresponds to a power exchange with the grid. In the case of an off-grid system, a production curtailment or a load shedding is required. The instantaneous operational revenues we consider correspond to the financial impact of a surplus or lack of production. The reward function r_t is a linear function of the power balance δ_t and, because the price β at which the energy surplus can be sold to the grid usually differs from the retail price k to buy electricity from the grid, the definition of the reward function at time step $t \in \mathcal{T}$ depends of the sign of δ_t:

$$r_t = \begin{cases} \beta\ \delta_t \Delta t & \text{if } \delta_t \geqslant 0\,, \\ k\ \delta_t \Delta t & \text{otherwise.} \end{cases} \tag{5.32}$$

Using Equations 5.29, 5.30, and 5.31, the reward function can be expressed as a function of the system variables:

$$r_t = \begin{cases} \text{if } -p_t^B - p_t^{H_2} - c_t + \eta_t^{PV} x_t^{PV} i_t \geqslant 0: \\ \quad \beta\ (-p_t^B - p_t^{H_2} - c_t + \eta_t^{PV} x_t^{PV} i_t)\ \Delta t, \\ \text{otherwise:} \\ \quad k\ (-p_t^B - p_t^{H_2} - c_t + \eta_t^{PV} x_t^{PV} i_t)\ \Delta t. \end{cases} \tag{5.33}$$

5.3 OPTIMISATION

In this section, we detail how to implement the LEC version of Problems 1, 2, and 3, to obtain an optimal solution using mathematical programming techniques. Even though the formalization of the problem includes non-linear relations (e.g., Equations 5.18, 5.19, and 5.33), we show how to obtain a linear program by using auxiliary variables. The presented approach assumes the following conditions:

- a single candidate technology is considered for each device type (i.e., $J = L = M = 1$);
- the lifetime of the devices is at least as long as the considered time-horizon (i.e., $L^{PV}, L^B, L^{H_2} \geqslant T$) and the aging of the devices can thus be ignored;
- the whole trajectory $E_1, \ldots, E_T$ of environment vectors is known at the time of operation (i.e., when minimizing G_T).

These assumptions could be removed at the price of a non-linear optimization (possibly increasing significantly the computation time).

5.3.1 *Optimal operation over a known trajectory of the exogenous variables*

We first consider the implementation as a linear program of Problem 1 with G_T defined by Equation 5.27. The output of this program is

the optimal sequence of control actions $\mathbf{a}_t^{(o)} = (p_t^{H_2}, p_t^B)$ and the corresponding minimal value of G_T over the considered time-horizon T. Before writing the optimization model, we introduce, $\forall t \in \mathcal{T}$, the following auxiliary variables:

$$p_t^{B,+}, p_t^{B,-}, p_t^{H_2,+}, p_t^{H_2,-}, \delta_t^+, \delta_t^- \in \mathbb{R}^+,$$

$$\text{such that } \begin{cases} p_t^B = p_t^{B,+} - p_t^{B,-} \\ p_t^{H_2} = p_t^{H_2,+} - p_t^{H_2,-} \\ \delta_t = \delta_t^+ - \delta_t^- \end{cases}$$

which allows the use of the adequate efficiency factor (i.e., η or ξ) and price (i.e., k or β) depending on the direction of the power flows. The overall linear program $\mathcal{M}_{op}$, having as parameters the time-horizon T, the time step Δt, the number of years n spanned by the time-horizon, the sets $\tau_1, \ldots, \tau_n$ mapping years to time steps, the discount rate ρ, a trajectory $\mathbf{E}_1, \ldots, \mathbf{E}_T$ of the exogenous variables, and the time-invariant sizing state $\mathbf{s}^{(s)} = (x^{PV}, x^B, x^{H_2}, P^B, R^{H_2}, \eta^{PV}, \eta^B, \eta^{H_2}, \zeta^B, \zeta^{H_2}, r^B, r^{H_2})$ of the devices, can be written as:

$$\mathcal{M}_{op}(T, \Delta t, n, \tau_1, \ldots, \tau_n, r, \mathbf{E}_1, \ldots, \mathbf{E}_T, \mathbf{s}^{(s)}) = \min \frac{\sum_{y=1}^{n} \frac{M_y}{(1+\rho)^y}}{\sum_{y=1}^{n} \frac{\sum_{t \in \tau_y} c_t \Delta t}{(1+\rho)^y}} \tag{5.34a}$$

$$\text{s.t. } \forall y \in \{1, \ldots, n\}: \tag{5.34b}$$

$$M_y = \sum_{t \in \tau_y} \left(k\, \delta_t^- - \beta\, \delta_t^+ \right) \Delta t\,, \tag{5.34c}$$

$$\forall t \in \{1, \ldots, T\}: \tag{5.34d}$$

$$0 \leqslant s_t^B \leqslant x^B\,, \tag{5.34e}$$

$$0 \leqslant s_t^{H_2} \leqslant R^{H_2}\,, \tag{5.34f}$$

$$-P^B \leqslant p_t^B \leqslant P^B\,, \tag{5.34g}$$

$$-x^{H_2} \leqslant p_t^{H_2} \leqslant x^{H_2}\,, \tag{5.34h}$$

$$\delta_t = -p_t^B - p_t^{H_2} - c_t + \eta^{PV} x^{PV} i_t\,, \tag{5.34i}$$

$$p_t^B = p_t^{B,+} - p_t^{B,-}\,, \tag{5.34j}$$

$$p_t^{H_2} = p_t^{H_2,+} - p_t^{H_2,-}\,, \tag{5.34k}$$

$$\delta_t = \delta_t^+ - \delta_t^-\,, \tag{5.34l}$$

$$p_t^{B,+}, p_t^{B,-}, p_t^{H_2,+}, p_t^{H_2,-}, \delta_t^+, \delta_t^- \geqslant 0\,, \tag{5.34m}$$

$$s_1^B = 0, s_1^{H_2} = 0, \tag{5.34n}$$

$$\forall t \in \{2, \ldots, T\}: \tag{5.34o}$$

$$s_t^B = r^B s_{t-1}^B + \eta^B p_{t-1}^{B,+} - \frac{p_{t-1}^{B,-}}{\zeta^B}\,, \tag{5.34p}$$

$$s_t^{H_2} = r^{H_2} s_{t-1}^{H_2} + \eta^{H_2} p_{t-1}^{H_2,+} - \frac{p_{t-1}^{H_2,-}}{\zeta^{H_2}}\,, \tag{5.34q}$$

$$-\zeta^B s_T^B \leqslant p_T^B \leqslant \frac{x^B - s_T^B}{\eta^B}\,, \tag{5.34r}$$

$$-\zeta^{H_2} s_T^{H_2} \leqslant p_T^{H_2} \leqslant \frac{R^{H_2} - s_T^{H_2}}{\eta^{H_2}}\,. \tag{5.34s}$$

The physical limits of the storage devices are modeled by Constraints 5.34e to 5.34h, while the transition laws of their state correspond to Constraints 5.34p and 5.34q. Because of the absence of time step $T+1$, there is no guarantee that the charge levels that immediately follow the time-horizon are positive, which is why Constraints 5.34r and 5.34s ensure that the last action $\mathbf{a}_T^{(o)}$ is compatible with the last charge level of the devices. Finally, Constraints 5.34i and 5.34c respectively denote the power balance within the microgrid and the cost it induces on a yearly scale.

5.3.2 *Optimal sizing under optimal operation*

In Problem 2, the initial sizing of the microgrid becomes an output of the optimization model and the function G_0, here defined by Equation 5.26, integrates the objective function. We denote this new problem by $\mathcal{M}_{\text{size}}$, which is still a linear program:

$$\mathcal{M}_{\text{size}}(T,\Delta t,n,\tau_1,\dots,\tau_n,r,\mathbf{E}_0,\mathbf{E}_1,\dots,\mathbf{E}_T) = \min \frac{I_0 + \sum_{y=1}^{n} \frac{M_y}{(1+\rho)^y}}{\sum_{y=1}^{n} \frac{\sum_{t\in\tau_y} c_t \Delta t}{(1+\rho)^y}} \tag{5.35a}$$

$$\text{s.t. } I_0 = a_0^{PV} c_0^{PV} + a_0^{B} c_0^{B} + a_0^{H_2} c_0^{H_2}\,, \tag{5.35b}$$

$$(x^B, x^{H_2}, x^{PV}) = (a_0^B, a_0^{H_2}, a_0^{PV})\,, \tag{5.35c}$$

$$5.34b - 5.34s\,. \tag{5.35d}$$

This new model includes all the constraints from $\mathcal{M}_{\text{op}}$, as well as the definition of the sizing of the devices from the initial sizing decisions (i.e., Constraint 5.35c) and the expression of the initial investment as a function of these sizing decisions (i.e., Constraint 5.35b). Note that the value of physical properties of the devices other than variables x^B, x^{H_2}, x^{PV} is provided by the initial environment vector $\mathbf{E}_0$, which also provides the cost of the available technology for every device type.

5.3.3 *Robust optimization of the sizing under optimal operation*

The extension of linear program $\mathcal{M}_{\text{size}}$ to an optimization model $\mathcal{M}_{\text{rob}}$ that integrates a set $\mathbf{E} = \{(E_t^1)_{t=1\dots T}, \dots, (E_t^N)_{t=1\dots T}\}$ of candidate trajectories of the environment vectors (i.e., to the implementation of Problem 3) is straightforward and requires two additional levels of optimization:

$$\mathcal{M}_{\text{rob}}(T,\Delta t,n,\tau_1,\dots,\tau_n,r,\mathbf{E}_0,\mathbf{E}) = \min_{a_0^B, a_0^{H_2}, a_0^{PV}} \max_{i\in 1,\dots,N} \mathcal{M}_{\text{size}}(T,\Delta t,n,\tau_1,\dots,\tau_n,r,\mathbf{E}_0,\mathbf{E}_1^{(i)},\dots,\mathbf{E}_T^{(i)})\,. \tag{5.36}$$

This mathematical program cannot be solved using only linear programming techniques. In particular, the numerical results reported

further in this chapter relied on an exhaustive search approach to address the outer min max, considering a discretized version of sizing variables.

5.4 SIMULATIONS

This section presents case studies of the proposed operation and sizing problems of a microgrid. We first detail the considered technologies, specify the corresponding parameter values, and showcase the optimal operation of a fixed-size microgrid. The optimal sizing approaches are then run using realistic price assumptions and using historical measures of residential demand and of solar irradiance with $\Delta t = 1h$. By comparing the solutions for irradiance data of both Belgium and Spain, we observe that they depend heavily on this exogenous variable. Finally, we compare the obtained LEC values with the current retail price of electricity and stress the precautions to be taken when interpreting the results.

5.4.1 *Technologies*

In this subsection, we describe the parameters that we consider for the PV panels, the battery and the hydrogen storage device. The physical parameters are selected to fit, at best, the state-of-the-art manufacturing technologies, and the costs that we specify are for self-sufficient devices, i.e., including the required converters or inverters to enable their correct operation.

PV PANELS. The electricity is generated by converting sunlight into direct current (DC) electricity using materials that exhibit the photovoltaic effect. Driven by advances in technology as well as economies of manufacturing scale, the cost of PV panels has steadily declined and is about to reach a price of $1€/W_p$ with inverters and balance of systems included [Ossenbrink et al., 2013]. The parameters that are taken into account in the simulations can be found in Table 5.1.

Parameter	Value
c^{PV}	$1€/W_p$
η^{PV}	18%
L^{PV}	20 years

Table 5.1: Characteristics used for the PV panels.

BATTERY The purpose of the battery is to act as a short-term storage device; it must therefore have good charging and discharging

efficiencies as well as enough specific power to handle all the short-term fluctuations. The charge retention rate and the energy density are not major concerns for this device. A battery's characteristics may vary due to many factors, including internal chemistry, current drain and temperature, resulting in a wide range of available performance characteristics. Compared to lead-acid batteries, $LiFePO_4$ batteries are more expensive but offer a better capacity, a longer lifetime and a better power density [Chih-Chiang and Zong-Wei, 2010]. We consider this latter technology and Table 5.2 summarizes the parameters that we deem to be representative. $LiFePO_4$ batteries are assumed to have a power density that is sufficient to accommodate the instantaneous power supply of the microgrid. It is also assumed to have a charging efficiency (η^c) and discharging efficiency (ζ_0^B) of 90% for a round trip efficiency of 81%. Finally, we consider a cost of 500 € per usable kWh of storage capacity (c^B).

Parameter	Value
c^B	500 €/kWh
η_0^B	90%
ζ_0^B	90%
P^B	> 10kW
r^B	99%/month
L^B	20 years

Table 5.2: Data used for the $LiFePO_4$battery.

HYDROGEN STORAGE DEVICE The long-term storage device must store a large quantity of energy at a low cost while its specific power is less critical than that for the battery. In this chapter we will consider a hydrogen-based storage technology composed of three main parts: (i) an electrolyzer that transforms water into hydrogen using electricity (ii) a tank where the hydrogen is stored (iii) a fuel cell where the hydrogen is transformed into electricity (note that a (combined heat and) power engine could be used instead). This hydrogen storage device is such that the maximum input power of the fuel cell before losses is equal to the maximum output power of the electrolyzer after losses. The considered parameters are presented in Table 5.3.

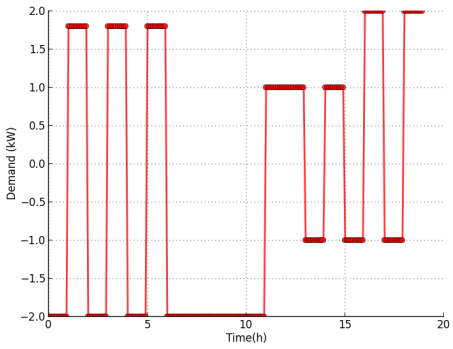
5.4.2 *Optimal operation*

An example of output of the optimal operation program $\mathcal{M}_{op}$ in Figure 5.2b illustrates well the role of each storage device. The figure sketches the evolution of the charge levels of the battery and of the hydrogen storage device when facing the net demand defined in Figure 5.2a. In this example, the battery has a capacity of 3kWh and

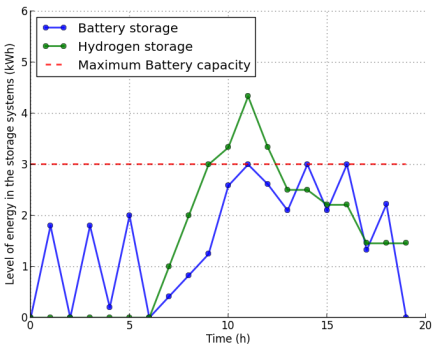
Parameter	Value
c^{H_2}	14 €/W_p
$\eta_0^{H_2}$	65%
$\zeta_0^{H_2}$	65%
r^{H_2}	99%/month
L^{H_2}	20 years
R^{H_2}	∞

Table 5.3: Data used for the Hydrogen storage device.

the hydrogen storage device has a power limit of 1kW. The role of each storage device is clear as we observe that the battery handles the short fluctuations, while the hydrogen device accumulates the excesses of production on a longer time-scale. Overall, since the production is higher than the consumption by a significant margin, the optimization problem is not constrained and hydrogen is left in the tank at the end of the simulation.



(a) Net demand (negative demand represents a production higher than the consumption)



(b) Optimal operation of the storage devices

Figure 5.2: Figure (b) shows the evolution of the charge levels within a microgrid that faces the given net demand given in Figure (a).

5.4.3 *Production and consumption profiles*

In this subsection, we describe the PV production profiles and the consumption profiles that will be used in the remaining simulations.

5.4.3.1 *PV production*

Solar irradiance varies throughout the year depending on the seasons, and it also varies throughout the day depending on the weather and the position of the sun in the sky relative to the PV panels. Therefore, the production profile varies strongly as a function of the geographical area, mainly as a function of the latitude of the location. The two cases considered in this chapter are a residential consumer of electricity in the south of Spain and in Belgium. The main distinction between these profiles is the difference between summer and winter PV production. In particular, production in the south of Spain varies with a factor 1:2 between winter and summer (see Figure 5.3) and changes to a factor of about 1:5 in Belgium or in the Netherlands (see Figure 5.4).

5.4.3.2 *Consumption*

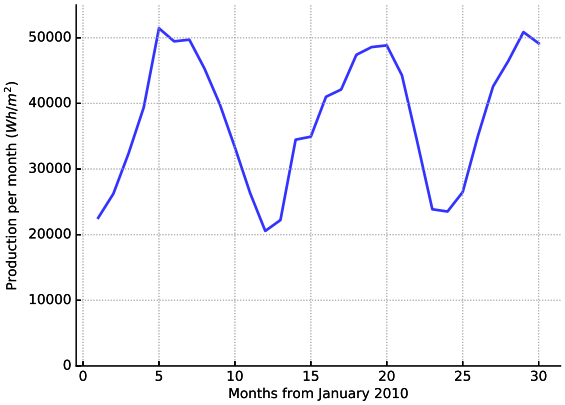
A simple residential consumption profile is considered with a daily consumption of 18kWh. The profile can be seen on Figure 5.5. In a more realistic case, particular precautions should be taken in the case of high consumption peaks to ensure that the battery will be able to handle large power outputs. In addition, the consumption profile may have higher daily average consumption during winter than in summer, which may substantially affect the sizing and operation solutions.

5.4.4 *Optimal sizing and robust sizing*

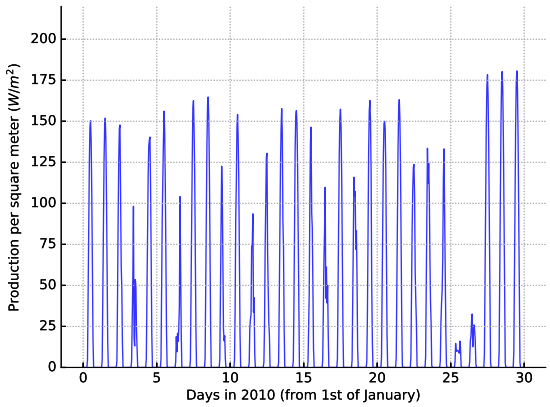
For the optimal sizing under optimal operation of the microgrid, as defined by Problem 2, we use a unique scenario built from the data described in Section 5.4.3 for the consumption and production profiles. Since the available data are shorter than the time-horizon, we repeat them so as to obtain a twenty-year-long time-horizon. In the following we make the hypothesis that $\beta = 0$ €/kWh.

For the robust optimization of the sizing, we refer to the Problem 3. This approach requires the selection of a set of different environment trajectories and, for computational purposes, to discretize the sizing states. The three different scenarios considered are the following:

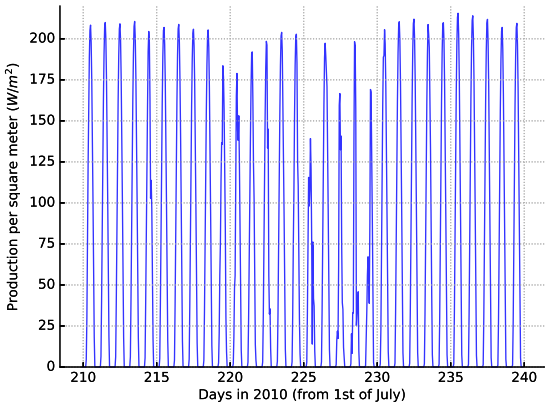
- The production is 10% lower and the consumption is 10% higher than the representative residential production/consumption profile.



(a) Total energy produced per month

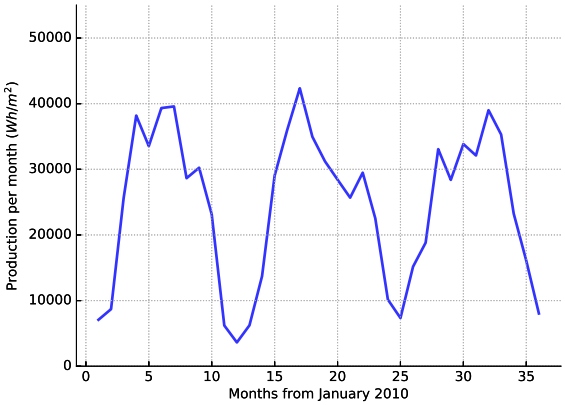


(b) Example of production in winter

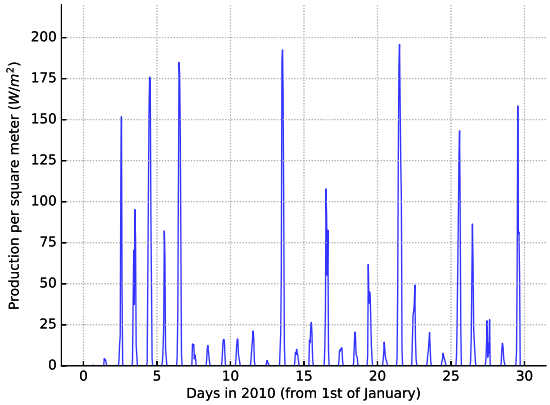


(c) Example of production in summer

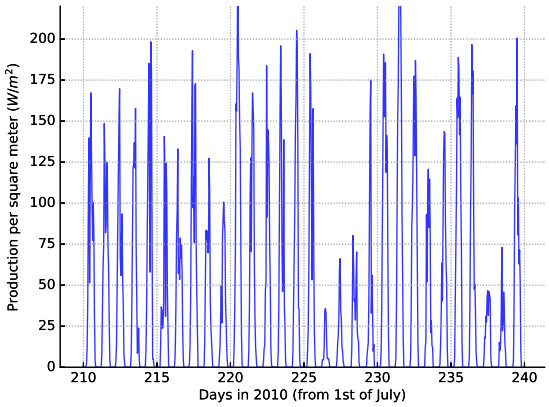
Figure 5.3: Simulated production of PV panels in the South of Spain (Data from Solargis [Šúri et al., 2011] for the solar platform of Almeria in Spain).



(a) Total energy produced per month



(b) Example of production in winter



(c) Example of production in summer

Figure 5.4: Measurements of PV panels production for a residential customer located in Belgium.

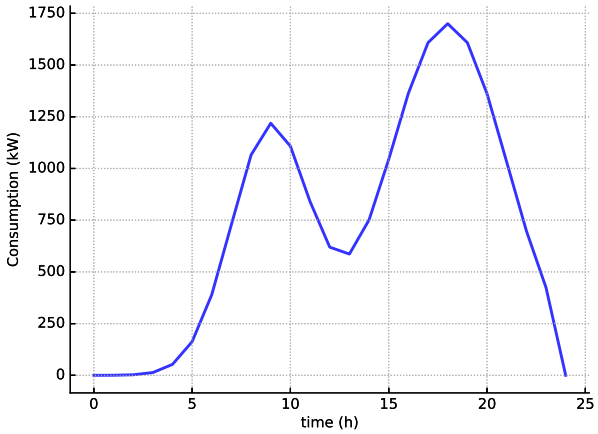


Figure 5.5: Representative residential consumption profile.

- The production and the consumption are conform to the representative residential production/consumption profile (scenario used in the non-robust optimisation)
- The production is 10% higher and the consumption is 10% lower than the representative residential production/consumption profile.

To build the discretized sizing states we start by solving Problem 2 on the mean scenario. For our simulations we then select all possible variations compared to the sizing of each variable x^B, x^{H_2} and x^{PV} by +0%, +10% and +20%. This leaves us with 27 possible sizings that are used to build the discretized sizing space. Equation 5.36 is solved by performing an exhaustive search over this set of potential sizings so as to obtain the robust LEC.

5.4.4.1 *The Spanish case*

We first considered a residential consumer of electricity located in Spain. For different values of costs k incurred per kWh not supplied within the microgrid, we performed the optimal sizing and the robust-type optimization schemes described above. We reported the obtained LEC in Figure 5.6. We observed the following : (i) for a retail price of 0.2€/kWh, the residential consumer of electricity benefits from a LEC of slightly more than 0.10€/kWh; (ii) in the fully off-grid case, the microgrid is still more profitable than buying electricity at all times from the utility grid for all configurations as long as k is lower than approximately 3€/kWh (i.e., with a value of loss load smaller than 3 €/kWh, it is preferable to go fully off-grid if the only other alternative is to buy all the electricity from the grid); (iii) due to the relatively low inter-seasonal fluctuations (compared to Belgium for instance (see later)) investing in a hydrogen storage system is not actually profitable for low values of k.

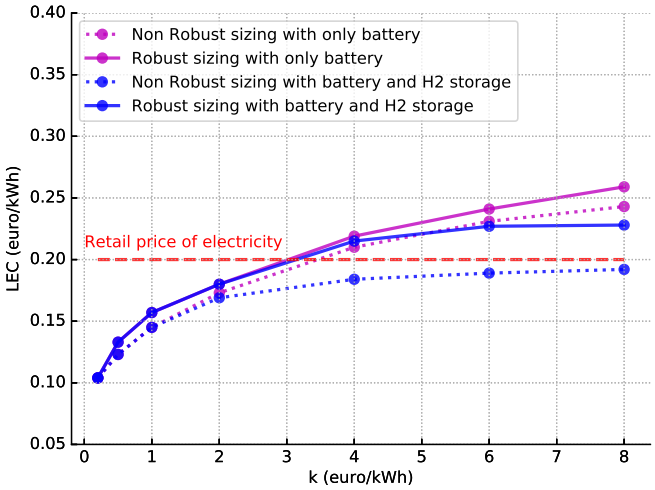


Figure 5.6: LEC ($\rho = 2\%$) in Spain over 20 years for different investment strategies as a function of the cost incurred per kWh not supplied within the microgrid.

5.4.4.2 *The Belgian case*

We then considered a residential consumer of electricity located in Belgium and we reported the obtained LEC for different values of k. As can be seen from Figure 5.7, a residential consumer of electricity in Belgium has incentives to invest in his own microgrid system (at least PV panels) since the obtained LEC while operating in parallel with the main utility grid at a retail price of 0.2€/kWh gives the residential consumer of electricity a lower electricity price than buying it from the grid at all times. With the current state of the technology however, it is not yet profitable for a residential consumer of electricity in Belgium to go fully off-grid in the considered setting since they would then suffer from a higher overall cost. Contrary to the results observed for Spain, in Belgium there is an important potential gain in combining both short-term and long-term energy storage devices. This is due to the critical inter-seasonal fluctuations of PV electrical production in Belgium.

We also investigate how the LEC evolves as a function of the price decrease of the elements in the microgrid. Figure 5.8 shows the reported LEC as a function of a uniform price decrease of the elements of the microgrid while assuming a value of loss load of 0.2€/kWh and a robust sizing. It is shown that when the prices of constitutive elements of the microgrid are less than half of those given in Tables 5.1,5.2 and 5.3, the business case for a fully off-grid microgrid in Belgium may actually become cost-effective.

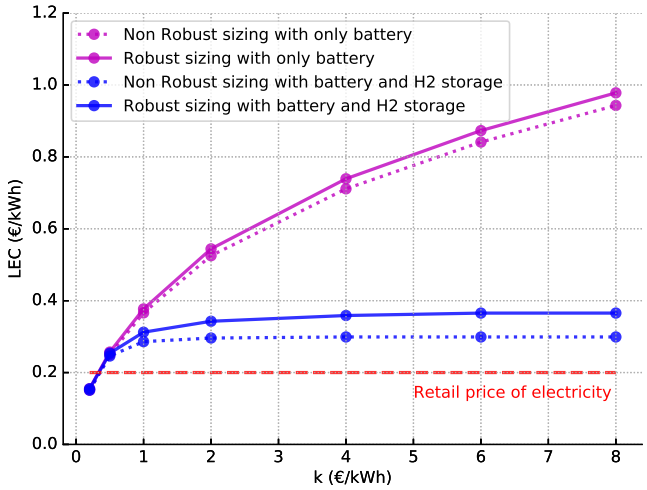


Figure 5.7: LEC ($\rho = 2\%$) in Belgium over 20 years for different investment strategies as a function of the cost incurred per kWh not supplied within the microgrid.

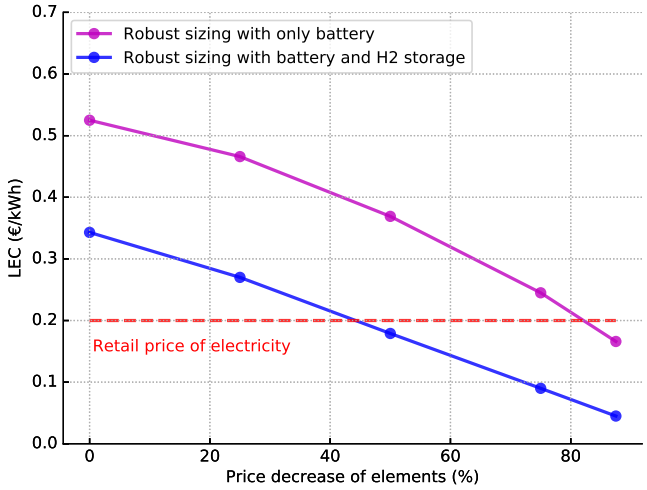


Figure 5.8: LEC ($\rho = 2\%$) in Belgium over 20 years for a value of loss load of 2€/kWh as a function of a uniform price decrease for all the constitutive elements of the microgrid.

5.5 CONCLUSION

This chapter has proposed a novel formulation of electrical microgrids featuring PV, long-term (hydrogen) and short-term (batteries) storage devices. This formulation relied on different assumptions concerning the components. The main hypotheses are that : (i) we considered one type of battery with a cost proportional to its capacity and one type of hydrogen storage with a cost proportional to the

maximum power flows, (ii) for the storage devices, we considered a charging and a discharging dynamics with constant efficiencies (independent of the power) and without aging (only limited lifetime), and (iii) for the PV panels, we considered a production to be proportional to the solar irradiance, also without any aging effect on the dynamics (only limited lifetime). In that context and with the hypothesis that the future consumption and production were known, we managed to set up an algorithm using linear programming for optimally sizing and operating microgrids under some (potentially robust) hypotheses on the surrounding environment. The approach has been illustrated in the context of Belgium and Spain, for which we have evaluated the values of the LEC and compared it with the cost of electricity from traditional electricity networks. Experimental results have shown that there was an important benefit in combining batteries and hydrogen-based storage, in particular when (i) there was an important difference in electricity production from the PV panels between summer and winter (e.g., in Belgium) and (ii) the cost for interruption (value of loss load) in the supply was high.

In the next chapter, we will relax the assumption that the future consumption and production are known when computing the operation. This will provide a realistic setting where the main challenge is to determine a real-time operation under uncertainty. One important question that will be answered concerns the additional costs subsequent to the relaxation of the hypothesis that the future production and future consumption are known at the time of operation.

6

DEEP REINFORCEMENT LEARNING SOLUTIONS FOR ENERGY MICROGRIDS MANAGEMENT

6.1 INTRODUCTION

Energy microgrids face a dual stochastic-deterministic structure: one of the main challenges to meet when operating microgrids is to find storage strategies capable of handling uncertainties related to future electricity production and consumption; besides this, a characteristic of microgrids is that their dynamics deterministically react to storage management actions.

In this chapter, we propose to design a storage management strategy that 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. This setting 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). In this context, we propose to specifically design a deep RL algorithm (based on the DQN algorithm introduced in Section 2.3.4) for approximating the optimal storage strategy through interaction with the environment. Unlike [Kuznetsova et al., 2013], our specific deep neural network architecture is built upon a large, continuous, non-handcrafted feature space that uses convolutional layers to extract meaningful features from the time series. Compared to the approach in [Mnih et al., 2015], we propose a validation strategy that 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 errors associated with scenarios aggregation in stochastic programming ([Mohammadi et al., 2014]).

This chapter is organized as follows: Section 6.2 introduces the control problem of the microgrid management. Section 6.3 describes the deep RL framework as well as empirical results corresponding to the case of a residential customer located in Belgium. Section 6.4 concludes this chapter.

6.2 CONTROL PROBLEM OF THE MICROGRID MANAGEMENT

We consider the case of a residential electricity prosumer (i.e. both consumer and producer) located in Belgium and operating an off-grid microgrid. The microgrid model and the microgrid parameters considered in this chapter are the same as the ones provided in Chapter 5, except for a few elements explicitly stated hereafter.

First, the consumption profile keeps an average consumption of 18kWh/day as in Chapter 5, but the profile is varied randomly by a factor of $\pm 25\%$ for each day (see Figure 6.1). This slightly different setting is chosen to demonstrate, without ambiguity, that the DQN algorithm is able to handle uncertainty coming from the consumption profile. Concerning the production profile, it is directly taken from Chapter 5, which already presents strong variations.

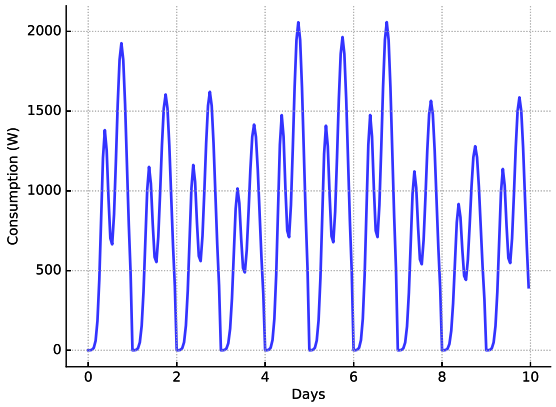


Figure 6.1: Representative residential consumption profile.

Second, a careful observation of the problem allows removing one degree of freedom in the action space (without introducing bias), thus simplifying the control problem. The only possible action considered in the control scheme relates to how the hydrogen storage device is controlled ($p_t^{H_2}$), while the control of the battery (p_t^B) follows deterministically by avoiding any direct value of loss load (except when the battery is at its lowest allowed level) and by avoiding wasting energy (except when the battery is full). As illustrated in Figure 6.2, we consider three discretized actions $a_t \in \mathcal{A}$ for the hydrogen storage device: (i) charge it at maximum rate, (ii) keep it idle or (iii) discharge it at maximum rate. This discretization is sufficiently flexible and without any major drawbacks since the time steps are sufficiently small ($\Delta t = 1h$).

Third, the instantaneous reward signal r_t is obtained by adding the revenues generated by the hydrogen production r^{H_2} and the penalties r^- due to the value of loss load:

$$r_t = r(a_t, d_t) = r^{H_2}(a_t) + r^-(a_t, d_t), \tag{6.1}$$

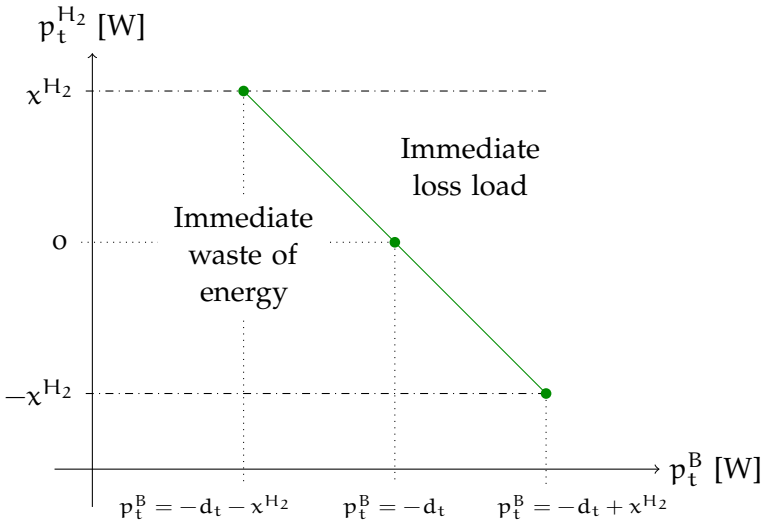


Figure 6.2: Sketch of the simplification of the control space. The green lines are the possible control actions where one degree of freedom has been removed and the green dots are the considered discretized actions.

where, as noted in the previous chapter, d_t denotes the net electricity demand, which is the difference between the local consumption c_t and the local production of electricity ψ_t. The penalty r^- is proportional to the total amount of energy that was not supplied to meet the demand, with the cost k incurred per kWh not supplied within the microgrid set to 2 €/kWh (corresponding to a value of loss load). The revenues (resp. cost) generated by the hydrogen production r^{H_2} is proportional to the total amount of energy transformed (resp. consumed) in the form of hydrogen, where the revenue (resp. cost) per Wh of hydrogen produced (resp. used) is set to 0.1 €/kWh. This setting can be related in practice to a microgrid having access to a hydrogen pipeline or having access to the possibility of selling (resp. buying) filled hydrogen cylinders. Note that explicitly giving a revenue (resp. cost) at each time step for the hydrogen production (resp. consumption) allows providing direct feedback to the agent about the interest of long-term storage which is a form of reward shaping; see, e.g., [Ng et al., 1999].

Finally, we consider a fixed sizing of the microgrid (that corresponds to the robust sizing provided in Chapter 5). 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 $x^{PV} = 12kW_p$.

6.3 APPLYING DEEP REINFORCEMENT LEARNING FOR MANAGING MICROGRIDS

Operating the microgrid is formalized as a partially observable Markov decision process, where the microgrid is considered as an agent that interacts with its environment. The dynamics is given by the following equation:

$$H_{t+1} = f(H_t, a_t, w_t), \tag{6.2}$$

where for all t, $H_t \in \mathcal{H}$ is a history of observations up to time t, the action a_t is an element of the action space $\mathcal{A}$ and the random disturbance w_t is an element of the disturbance space $\mathcal{W}$ generated by a probability distribution $w_t \sim P(.|H_t)$.

We define the γ-discounted optimal Q-value function as:

$$Q^*(\phi(H), a) = \max_{\pi} \mathop{\mathbb{E}}_{w_t, w_{t+1}, \dots} \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | \phi(H_t) = \phi(H), a_t = a, \pi \right]$$

and we propose to approximate Q^* using a DQN algorithm as presented in Section 2.3.4.

6.3.1 *Splitting the times series to avoid overfitting*

We consider a case where the agent is provided with two years of actual past realizations of the consumption (c_t) and the production (ψ_t). 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 has two purposes:

- During training, it is used at each epoch [1] to estimate how well the policy performs on the undiscounted objective M_y, and it selects the best (approximated) Q-network denoted $\widetilde{Q}^*$ before overfitting (by selecting a discount factor lower than the maximum and by using early stopping). It also has the advantage of picking up the Q-network at an epoch less affected by instabilities.
- It has been used for selecting via an informal search (i) the mapping ϕ, (ii) the structure of the neural network and (iii) the values of all other hyperparameters.

The selected trained Q-network is then used in a test environment ($y = 3$) to provide an independent estimation of how well the resulting policy performs.

1 An epoch is defined as the the set of all iterations required to go through the whole year of the exogenous time series c_t and ψ_t (8760 steps). Each iteration is made up of a transition of one time-step in the environment as well as a gradient step of all parameters θ of the Q-network.

6.3.2 *Mapping* ϕ

Three different cases are considered:

1. A base case with minimal information available to the agent: the mapping $\phi(H)$ is made up of

$$\phi(H_t) = \left[[c_{t-h^c}, \ldots, c_{t-1}], [\psi_{t-h^p}, \ldots, \psi_{t-1}], s_t^B\right],$$

   where $h^c = 12h$ and $h^p = 12h$ are the lengths of the time series considered for the consumption and production, respectively, and s_t^B is, as in the previous chapter, the level of energy in the battery.

2. A case where additional information on the season is provided:

$$\phi(H_t) = \left[[c_{t-h^c}, \ldots, c_{t-1}], [\psi_{t-h^p}, \ldots, \psi_{t-1}], s_t^B, \zeta_s\right],$$

   where ζ_s is the smallest number of days to the summer solstice (21^{st} of June).

3. A case where accurate production forecasting is available:

$$\phi(H_t) = \left[[c_{t-h^c}, \ldots, c_{t-1}], [\psi_{t-h^p}, \ldots, \psi_{t-1}], s_t^B, \zeta_s, \rho_{24}, \rho_{48}\right],$$

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

6.3.3 *Neural network architecture*

We propose a neural network architecture where the inputs are provided by $\phi(H_t)$ (normalized into [0,1]), and where the outputs represent the Q-values for each discretized action.

The neural network processes the 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 combination of the output of the convolutions and the non-time series inputs is then followed by two fully-connected layers with 50 and 20 neurons. The activation function used is the Rectified Linear Unit (ReLU) except for the output layer where no activation function is used. A sketch of the structure of the neural network is provided in Figure 6.3.

6.3.4 *Training*

By starting with a random Q-network with output values close to zero, two different processes happen at each iteration:

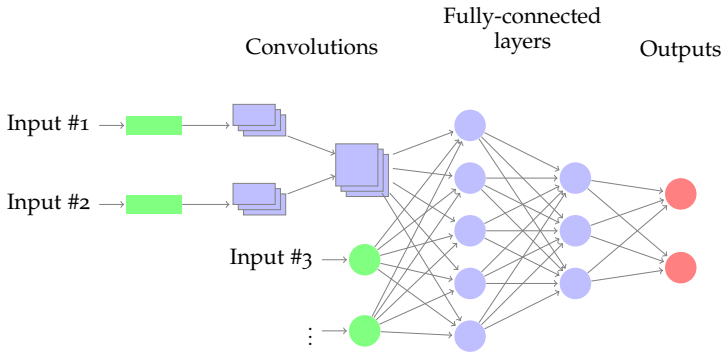


Figure 6.3: Sketch of the structure of the neural network architecture. The neural network processes the time series using a set of convolutional layers. The output of the convolutions and the other inputs are followed by fully-connected layers and the ouput layer. Architectures based on LSTMs instead of convolutions obtain similar results, and the reader is welcome to experiment with the source code provided in Appendix 6.5.

- An action is selected by following an ϵ-greedy policy s.t. the policy $\pi(s) = \max_{a \in \mathcal{A}} Q(\phi(H), a; \theta_k)$ is selected with a probability $1 - \epsilon$, and a random action (with uniform probability over actions) is selected with probability ϵ (we use decreasing value of ϵ over time). The replay memory is filled with the selected action, as well as the subsequent reward and observation.
- The update given in Eq. 2.11 is performed.

During the validation and test phases, the policy

$$\pi(s) = \max_{a \in \mathcal{A}} Q(\phi(H), a; \theta_k)$$

is applied (with $\epsilon = 0$).

As discussed in Chapter 4, we use an increasing discount factor along with a decreasing learning rate through the learning epochs in order to enhance learning performance. The hyperparameters used and the source code are provided in Appendix 6.5.

6.3.5 *Results and discussions*

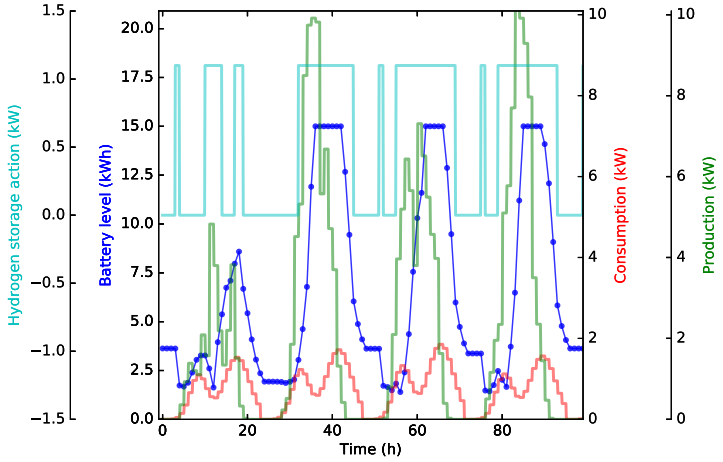
The typical behavior of the selected policy in the case with minimal information is illustrated in Figure 6.4 on the test data. Since the microgrid has no information about the future, it has to make a tradeoff between two objectives:

- it has to maintain a sufficient reserve in the short-term storage device to be able to face the consumption without suffering (much) loss load, and
- it also has to avoid wasting energy (when the short-term storage is full) by storing energy in the long-term storage device whenever possible.

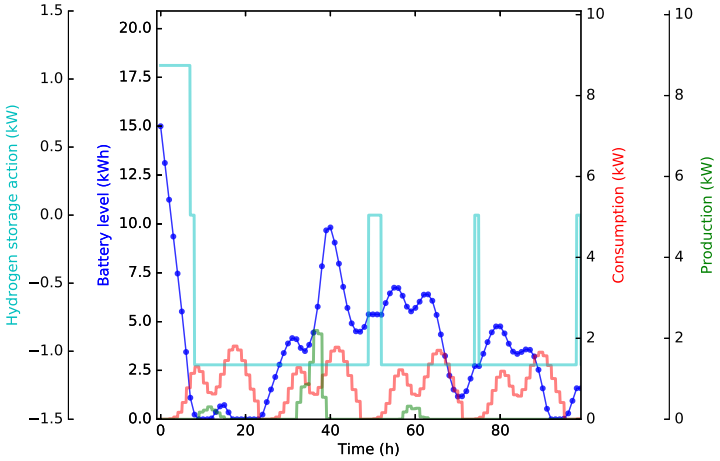
We now investigate the effect of providing additional information to the agent. The typical behavior of the policy in the case where accurate production forecasting is illustrated in Figure 6.5 on the test data. As compared to the previous case, it can be seen that it is able to manage the level of energy in the battery storage even more efficiently. When the production will be low for the next days, the microgrid aims to keep a larger quantity of energy in the battery, while when the production for the next days will be important, it tends to require a lower level of energy to be stored in the battery early in the morning.

We report in Figure 6.6a the operational revenue on the test data $M_y^{\pi_{\tilde{Q}^*}}$ for the three cases discussed in Section 6.3.2 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. This is due to the fact that the operation is more challenging in such cases and that 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 improve the policy, such as accurate information about the production profile. Similarly, additional data on the consumption profile would help further improve the policy $\pi_{\tilde{Q}^*}$. This data could, for instance, give information about the current day of the week in order to model the case where a residential customer would consume, on average, more energy during particular days (e.g., during the week-end).

The LEC obtained as a function of a unique percentage of the initial sizings x^B, x^{H_2}, x^{PV} is also reported in Figure 6.6b. 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.

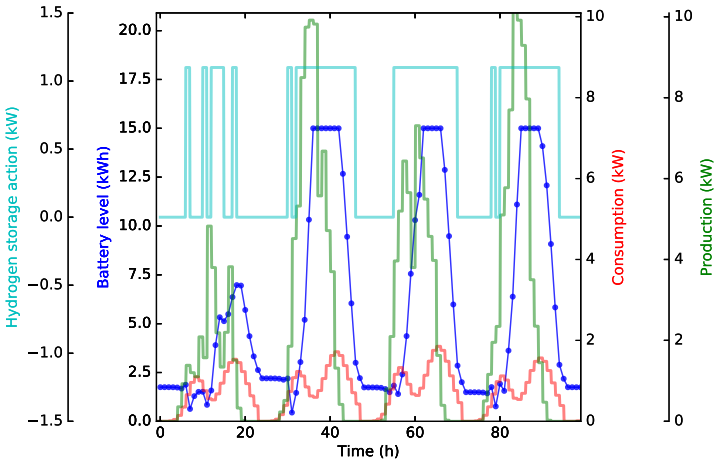


(a) Typical policy during summer

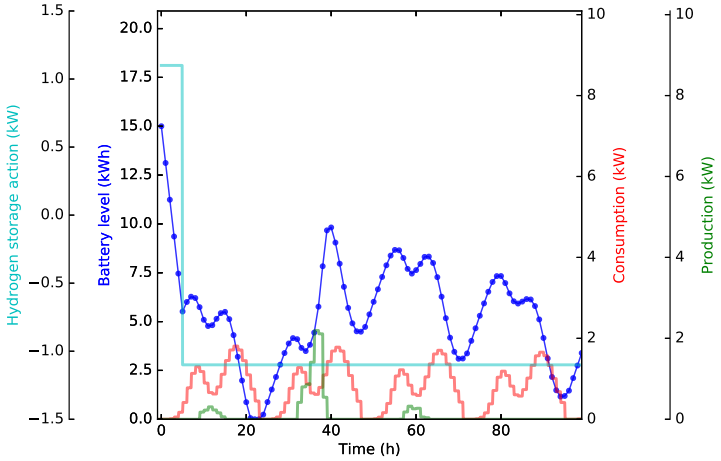


(b) Typical policy during winter

Figure 6.4: Illustration of the policy with minimal information available to the agent (test data).

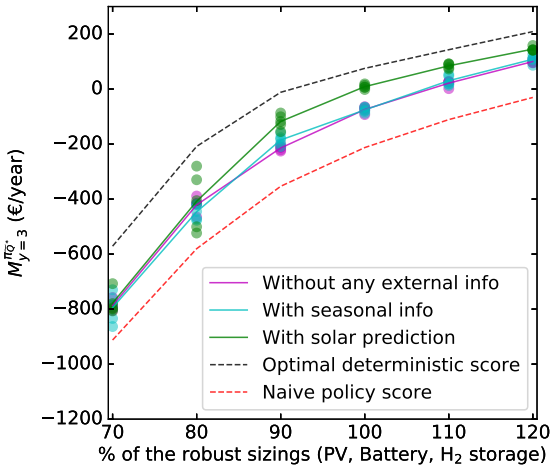


(a) Typical policy during summer

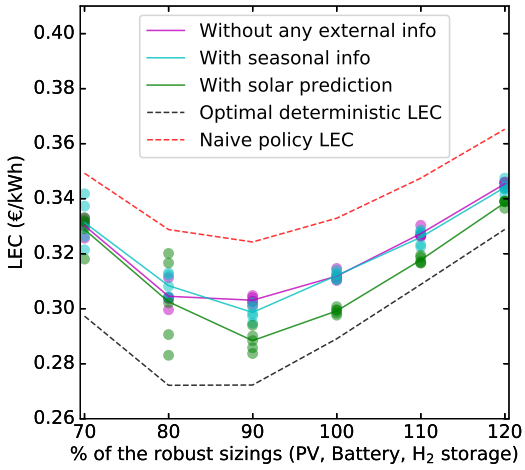


(b) Typical policy during winter

Figure 6.5: Illustration of the policy with accurate production forecast available to the agent (test data).



(a) Operational revenue



(b) LEC

Figure 6.6: Operational revenue and LEC (test data) function of the sizings of the microgrid. The optimal deterministic operation is obtained by solving the problem with the assumption of perfect knowledge of the whole future, using the method described in Chapter 5. The Naive policy operation is obtained by optimizing the thresholds at which the hydrogen storage is charged or discharged based on the level of energy in the battery (through grid search on rollouts in the validation environment).

How could the results be improved further?
With more computation time, many approaches similar to supervised learning could be used. Here are a few ideas:

- using k-fold cross validation,
- using data augmentation by generating trajectories similar to the consumption and production profiles,
- investigating different neural network structures such as combining LSTM's and convolutions, or
- performing Bayesian optimization (or a systematic grid search) on different hyperparameters.

6.4 CONCLUSION

This chapter has introduced a deep RL architecture for addressing the problem of operating an electricity microgrid. The only required assumptions, rather realistic, are that (i) the dynamics of the different constituting elements of the microgrid are known and that (ii) past time series providing the weather-dependent PV production and the consumption within the microgrid are available. In that context, the environment considered in this paper face a specific structure with a dynamics that is partly deterministic and partly depending on the exogenous stochastic time-series; the proposed approach is thus original since the training and validation phases make use of this particular structure to build a policy (via a Q-network). Experimental results illustrate the fact that the neural network representation of the value function efficiently generalizes the policy to situations corresponding to unseen configurations of solar irradiance and electricity consumption. These results have also shown that the additional cost as compared to the optimal (which is obtained under the fully deterministic setting described in the previous chapter) could be kept relatively small thanks to the deep RL techniques described in this chapter.

Future works could 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.

6.5 APPENDIX

PV production and consumption profiles as well as source code are available at https://github.com/VinF/deer. Documentation is available at http://deer.readthedocs.io/.

The different hyperparameters used are provided in Table 6.1.

Parameter	
Update rule	RMSprop ($\rho_{RMS} = 0.9$) [Tieleman, 2012]
Initial learning rate	α_0=0.0002
Learning rate update rule	$\alpha_{l+1} = 0.99\alpha_l$
RMS decay	0.9
Initial discount factor	γ_0=0.9
Discount factor update rule	$\gamma_{l+1} = \min(0.98, 1 - 0.99(1 - \gamma_l))$
Parameter of the ϵ-greedy	$\epsilon = max(0.3, 1 - \frac{k}{500.000})$
Replay memory size	1000000
Batch size	32
Freeze interval	1000 iterations
Initialization of the NN layers	Glorot uniform [Glorot and Bengio, 2010]

Table 6.1: Main parameters of the DQN algorithm. The integer k refers to the iteration number from the beginning of the training and the integer l refers to the epoch number.

CONCLUSION

7 CONCLUDING REMARKS

In this conclusion, we first summarize the contributions of this thesis, we then follow by the future work that these contributions highlight and we end up with a short discussion on some of the opportunities and risks of the future use of artificial intelligence.

7.1 CONTRIBUTIONS

Deep RL requires caution and understanding of its inner mechanisms in order to apply it successfully in different settings. In this thesis, we have contributed to the understanding of those mechanisms, and we have studied an application in the field of smartgrids. We hereafter summarize the main contributions of this thesis.

In chapter 2, we have provided a review of the domain of reinforcement learning with a focus on the recent developments in the field of deep RL. We have emphasized that deep learning has brought its generalization capabilities to the RL setting and, therefore, has provided a solution to the curse of dimensionality when working with time series, images, etc. We have discussed the different approaches (model-based, model-free) to the problem of deep RL and have discussed their respective strengths depending on the problem, as well as the possibility of combining them. We have also discussed the current main challenges in RL, including how it can be used in real-world problems. We have also introduced the open-source library "DeeR" that has been initiated in the context of this thesis.

In chapter 3, we have brought contributions to the partially observable context (POMDP setting) where only limited data is available (batch RL). This batch setting is central in RL since even when additional data can be gathered (online setting), obtaining a performant policy from given data is part of the solution to an efficient exploration/exploitation tradeoff. In the batch POMDP setting, we have formalized a decomposition of the sub-optimality of deep RL in two terms: (i) an asymptotic bias that is independent of the quantity of data obtained and (ii) a term of overfitting that is a consequence of the fact that only limited data is available. In that context, we have provided theoretical and empirical contributions concerning the state representation in the context of finding a tradeoff capable of minimizing the overall sub-optimality; namely, we have emphasized that a guaranteed good state representation requires to efficiently discriminate the underlying state of the POMDP (to avoid a risk of a large asymptotic bias) while at the same time it should

avoid unnecessary complexity (to avoid a risk of a large overfitting). We have also investigated the effect of the function approximator and of the discount factor in light of this bias-overfitting tradeoff.

In chapter 4, we have focused on the case of the discount factor in a value iteration algorithm. Its study is interesting in that setting not only because of its effect on the bias-overfitting error but also because it plays a key role in the instabilities (and overestimations) of the value iteration algorithm. We have provided empirical results showing the interest of using an increasing discount factor to reduce errors and instabilities during the deep Q-learning iterations, while still targeting a low asymptotic bias. In addition, the effect of the learning rate and the possibility to fall in a local optimum have been illustrated.

In the last part of this thesis, we have discussed an application in the domain of smartgrids.

In chapter 5, we have formalized the problem of a microgrid featuring PV, long-term (hydrogen) and short-term (batteries) storage devices. Under hypotheses on the different components and under the hypothesis that the consumption and production scenarios were known, we have shown that linear optimization techniques can be used to solve the problem of both the optimal operation and the optimal sizing of the microgrid. In that context, we have shown that there was a potential important benefit in combining batteries and hydrogen-based storage as compared to only batteries, in particular when (i) the PV production between summer and winter was very different (e.g., a factor 5:1 in Belgium) and (ii) the cost for interruption (value of loss load) in the supply was high.

In chapter 6, we have used the microgrid model developed in chapter 5, but we have considered a realistic setting by relaxing the assumption that the future consumption and production were known when computing the operation. We have shown how deep RL techniques can be used to obtain a performant real-time operation in a stochastic and partially observable environment, and we were able to provide the sub-optimality in terms of costs (operation cost and LEC) due to the uncertainties related to future electricity production and consumption within the microgrid. In this chapter, we have applied many methodological elements studied in the first part of this thesis (introduction of a validation phase to obtain the best bias-overfitting tradeoff, increasing discount factor, etc.). In terms of future applications, this chapter has shown the potential of using deep RL in the domain of smartgrids. The work done in that chapter is also of practical interest outside the smartgrids domain. Indeed, it has been shown how to deal with sequential decision-making processes where part of the dynamics is well-known but where the dynamics depends on exogenous time series where only limited past realizations are available (e.g., trading market, weather-dependent decision-making tasks).

7.2 FUTURE WORK

Sequential decision-making remains an open field of research for which many theoretical, methodological and experimental challenges are still left unanswered. Within that domain, deep RL is a fast-growing approach that has already unlocked the possibility to tackle difficult problems. There is every reason to think that this development will continue in the coming years with more efficient algorithms and many new applications.

In terms of algorithmic work, one of the central questions that has been at the core of this thesis was how to deal with limited data. To this purpose, the new developments in the field of deep RL will surely develop the current trends of taking explicit algorithms and making them differentiable so that they can be embedded in a specific form of neural network and can be trained end-to-end. We therefore expect to see richer and smarter structures of deep RL algorithms that will be able to tackle an even wider range of applications than they currently do today. We also expect to see deep RL algorithms going in the direction of life-long learning where previous knowledge (in the form of pre-trained networks) can be embedded so as to increase performance and training time.

In terms of applications, many new areas will be impacted by the possibilities brought by deep RL. It is always difficult to predict the timelines for the different developments, but we believe that the current interest in deep RL is only the beginning of foreseeable profound transformations in information and communication technologies. In the context of smartgrids alone, the number of potential applications of deep RL are tremendous and goes beyond microgrids described in this thesis. An overview of many potential applications is given in [Glavic et al., 2017]:

- They are linked to different types of elements within the power system: subsystems controls, market-based high-level decision, planification of investments, etc.
- They serve different purposes: usual operation, restorative operations, emergency behaviors, etc.
- The time-scale of the decision problems ranges from fractions of seconds to years.

7.3 SOCIETAL IMPACT OF ARTIFICIAL INTELLIGENCE

Current developments in artificial intelligence (both for deep RL or in general for machine learning) come in the sequel of many tools brought by ICT technologies. As with all new technologies, this comes with different potential opportunities and threats for our society.

On the positive side, artificial intelligence promises great value to people and society. They have the potential to enhance the quality of life by automating tedious and exhausting tasks. They may improve education by providing adaptive content and keeping students engaged [Mandel et al., 2014]. They will surely improve public health with, for instance, intelligent clinical decision-making [Fonteneau et al., 2008]. They will also prevent millions of needless deaths with, among other things, self-driving cars [Bojarski et al., 2016]. They also have the possibility to help reducing greenhouse gas emissions, by e.g., optimizing traffic [Li et al., 2016]. And the list could go on.

But as with all powerful tools, they also bring societal, political and ethical challenges, raising the question of how they can be used for the benefit of all. We want to stress in this conclusion some of the potential issues that need to be dealt with, as we believe that these topics will require the active participation of the artificial intelligence community (with e.g., the "partnership on artificial intelligence" initiative).

We need to be careful that artificial intelligence is safe, reliable and predictable. As a simple example, to capture what we want an agent to do, we frequently end up, in practice, designing the reward function more or less arbitrarily. Often this works well, but sometimes it produces unexpected and potentially catastrophic behaviors. For instance, to remove a certain invasive species from an environment, one may design an agent that obtains a reward every time it removes one of these organisms. However, it is likely that to obtain the maximum cumulative rewards, the agent will learn to let that invasive species develop and only then would eliminate many of the invasive organisms, which is of course not the intended behavior.

The ethical use of artificial intelligence is also a concern. One of the reasons is that it will undoubtedly have an influence on people's everyday lives. As it is the case with most technologies, regulation should, at some point, ensure a positive impact of its usage.

Another issue is that artificial intelligence advances will probably have a strong influence on the economy and the job market. While these advances often come with the promise of bringing positive effects to our society, they also disrupt the job market and displace or modify some types of jobs. In fact, this type of concerns is not new and dates back, at least, to the industrial revolution (around early 19th century). Past experience can surely alleviate a few fears, such as the fear that jobs will simply disappear without new job creations. However, the artificial intelligence revolution may well have very specific consequences. A key challenge for the future is to make sure that the benefits of the developments in artificial intelligence and the wealth it generates do not deepen the inequalities in our society but instead are fairly shared.

We are still at the very first steps of artificial intelligence and the future is hard to predict. However, it is key that the potential issues

related to the development of artificial intelligence are progressively taken into consideration in public policies. If that is the case, we believe that artificial intelligence will have a positive impact on our society.

Part III

APPENDIX

A

USING APPROXIMATE DYNAMIC PROGRAMMING FOR ESTIMATING THE REVENUES OF A HYDROGEN-BASED HIGH-CAPACITY STORAGE DEVICE

This annex proposes a methodology to estimate the maximum revenue that can be generated by a company that operates a high-capacity storage device to buy or sell electricity on the day-ahead electricity market. The methodology exploits the Dynamic Programming (DP) principle and is specified for hydrogen-based storage devices that use electrolysis to produce hydrogen and fuel cells to generate electricity from hydrogen. Experimental results are generated using historical data of energy prices on the Belgian market. They show how the storage capacity and other parameters of the storage device influence the optimal revenue. The main conclusion drawn from the experiments is that it may be advisable to invest in large storage tanks to exploit the inter-seasonal price fluctuations of electricity.

A.1 INTRODUCTION

Developing sustainable energy systems is one of the most critical issues that today's society must address. Due to the fluctuating nature of the Renewable Energy Sources (RES) generation, fossil-fuel-based generation capacity is currently still needed to provide flexibility and to adequately cover peak demand. This issue can be partially addressed or mitigated in several ways CREG [2012]: (i) by diversifying the types of renewable energy sources to reduce the correlation between the amount of energy supplied by these sources, which lowers the risk of shortage of supply, (ii) by developing electricity storage capacity, (iii) by increasing the flexibility of the demand, to smooth out peak demand or (iv) by developing the electrical network since the variance in the energy supplied by renewable sources tends to decrease with the size of the zone on which they are collected Archer and Jacobson [2007]. This has been the main motivation for developing the European network in recent years. Note that authors have also reported that the variance in the energy supplied by renewables could be further decreased by building a global electrical grid that connects continents together Chatzivasileiadis et al. [2013], Chatzivasileiadis et al. [2014].

During recent years, storage has gradually become more and more profitable thanks to technological progress. Consequently, economic actors on the energy market are currently planning to invest additional funds in storage devices. Among the different storage tech-

nologies, pumped-storage hydroelectricity and batteries are currently among the most mature. Other technologies exist such as for example super capacities, energy conversion to natural gas, compressed air energy storage, flywheels, superconducting magnetic energy storage and storage of electricity in the form of hydrogen. This latter one seems to be particularly promising due to its capability to store large quantities of energy at a relatively low cost, and is therefore well suited for long-term storage Armaroli and Balzani [2011]. Additionally, the round trip efficiency of hydrogen-based storage devices is rather good. For example, the energy efficiency of an electrolyzer is around 80% and the one of a fuel cell is generally between 40% and 60%, which results in an overall round-trip efficiency of 35% up to 50%, with the potential to get an efficiency higher than 70% in hybrid fuel cell/turbine systems and more than 80% in Combined Heat and Power (CHP) systems.

However, before investing in such a hydrogen-based storage technology, a careful analysis of the return on investment needs to be carried out. Such an analysis implies, among others, to be able to estimate the revenues that can be generated by such a storage device on the power exchange markets, which is the focus of this annex. We will consider the case of a company that operates the hydrogen-based high-capacity storage device and makes money by buying or selling electricity on the day-ahead market. In such a context, the company has to decide on the day-ahead which amount of electricity to store or to generate for every market period. The main complexity of this decision problem originates from the fact that a decision to store or generate electricity at one specific market period may not only significantly impact the revenues that could be generated at other market periods of the day, but also the revenues that could be generated months ahead. As a result, long optimization horizons have to be considered for computing operation strategies for high-capacity storage devices.

The valuation of energy storage technologies on power markets has already received considerable attention in the scientific literature Eyer and Corey [2010]; Carmona and Ludkovski [2010]; Löhndorf and Minner [2010]; Mokrian and Stephen [2006]; Fleten and Kristoffersen [2007]; Salas and Powell [2013]. For example, reference Fleten and Kristoffersen [2007] proposes an approach based on mixed-integer programming for optimizing bidding strategies for hydropower. This approach can handle uncertainty in market prices and water inflows. However, the computational complexity of this technique grows very rapidly with the state/action space, which makes this approach unsuitable for estimating the revenues that can be generated by a storage capacity over an extended period of time. Another example is reference Salas and Powell [2013] where a methodology based on Approximate Dynamic Programming (ADP) is proposed for jointly optimizing in the day-ahead the trading of renewable energy and of the storage management strategies.

Before explaining the details of this approach, we will describe in Section A.2 the bid process for a typical day-ahead electricity market such as the Belgian electricity market and lay out in Section A.3 a mathematical model for energy storage utilizing hydrogen. A first formulation of our problem as a dynamic programming problem will be stated in Section A.4 where we assume that only the market prices of the next market day are known. Section A.5 specifies this formulation to the case where the market prices are assumed to be known over the whole optimization horizon and provides a fully specified algorithm that exploits this new formulation for computing the maximum operational revenue. The computational complexity of this algorithm is linear with respect to the cardinality of the discretized state space, the cardinality of the discretized action space, and the optimization horizon. Section A.6 provides experimental results computed from historical data gathered over the Belgian electricity market. Finally, Section A.8 concludes the annex.

A.2 OPTIMIZATION ON THE DAY-AHEAD ENERGY MARKET

Let us consider a power exchange market for the day-ahead trading of electricity, providing the market with a transparent reference price. Producers and retailers submit each day offers to the market operator for the day-ahead. An offer is defined by a volume and a limit price, and can span several market periods. The market clearing price is computed by the market operator at the intersection of the supply and the demand curves. The prices for electricity on the Belgian day-ahead market are determined via a blind auction with the possibility to define linked Block Orders that allow the execution of profile blocks to be subjected to the execution of other blocks. This possibility allows for the design of complex linked structures (i.e. families) that take into account the different possible price outcomes of the market clearing price. Figure A.1 shows the distribution of prices over the year 2013.

In this annex, we consider that the storage capacity is an agent which interacts with the electricity exchange market under the following assumptions:

- the evolution of the price of electricity does not depend on the behavior of this agent. This hypothesis is equivalent to assuming that the bids for supply or demand from the actor do not change the market clearing price significantly[1].
- the evolution of the prices is known when determining the agent behavior.

1 That effect is measured by the market resilience which is the price sensitivity due to an increase in offer or demand on the market. It is of the order of 5.10^{-3}€/MWh on the Belgian power exchange market Belpex [2014].

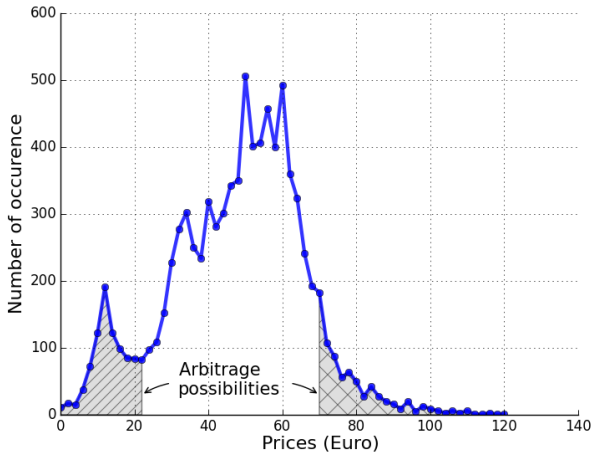


Figure A.1: Histogram of the price of electricity for the year 2013.

A.3 PROBLEM FORMALIZATION

Let us introduce a discrete-time system whose state variable is fully described by the amount of energy in the storage device. The state space $\mathcal{S}$ contains all possible states $s_{i,j} \in \mathcal{S}$, where the indices (i,j) refer to hour j during day i (in MWh). Let $\mathcal{A}$ be the set of possible actions and $\mathbf{a}_{i,j} \in \mathcal{A}$ the action taken at time (i,j). At every time step, an action $\mathbf{a}_{i,j} = [a_{i,j}^{GR}, a_{i,j}^{RG}] \in \mathcal{A}$ is applied on the system, where $a_{i,j}^{GR}$ is the amount of energy transferred into the storage (R) from the grid (G), and $a_{i,j}^{RG}$ is the amount of energy taken out of the storage (R) to the grid (G). The actions $a_{i,j}^{GR}$ and $a_{i,j}^{RG}$ are non-negative. The considered dynamics is defined over n_D days and n_H market periods ($n_H = 24$). We denote by $\mathtt{I}$ and $\mathtt{J}$ the sets of time indices:

$$\begin{aligned} \mathtt{I} &= \{0, ..., n_D - 1\}, \\ \mathtt{J} &= \{0, ..., n_H - 1\}. \end{aligned}$$

The system dynamics is given by the following equation:

$$\forall i \in \mathtt{I}, \forall j \in \mathtt{J}, \quad s_{i,j+1} = f(s_{i,j}, \mathbf{a}_{i,j}) \tag{A.1}$$

where we use the convention $s_{i,n_H} = s_{i+1,0}$ for any $i \in \mathtt{I}$. The notation $t_{i,j}$ is introduced as the time index corresponding to time $(i,j) \in \mathtt{I} \times \mathtt{J}$. This transition function can be rewritten as follows:

$$s_{i,j} = s_{i,0} + \sum_{t=t_{i,0}}^{t_{i,j-1}} (a_t^{GR} - a_t^{RG}), \quad \forall (i,j) \in \mathtt{I} \times \mathtt{J}. \tag{A.2}$$

At any time $(i,j) \in \mathtt{I} \times \mathtt{J}$, the following constraints have to be satisfied:

$$s_{i,0} + \sum_{t=t_{i,0}}^{t_{i,j-1}} (a_t^{GR} - a_t^{RG}) \leqslant R^c \tag{A.3}$$

$$s_{i,0} + \sum_{t=t_{i,0}}^{t_{i,j-1}} (a_t^{GR} - a_t^{RG}) \geqslant 0 \tag{A.4}$$

where R^c is the energy capacity of the device.

The bidding process occurs only once for each day $i \in \mathtt{I}$, which means that all actions taken on day $i+1$ are computed on day i. We denote by $\mathbf{s}_i \in \mathtt{S}$ the vector of states defined as $\mathbf{s}_i = [s_{i,0}, s_{i,1}, ..., s_{i,n_H-1}]$. We denote by $A_i \in \mathtt{A}$ the matrix of actions defined as follows: $A_i = \left[\mathbf{a}_i^{GR}; \mathbf{a}_i^{RG}\right]$ with

$$\mathbf{a}_i^{GR} = [a_{i,0}^{GR}, a_{i,1}^{GR}, ..., a_{i,n-1}^{GR}]$$

and

$$\mathbf{a}_i^{RG} = [a_{i,0}^{RG}, a_{i,1}^{RG}, ..., a_{i,n-1}^{RG}].$$

The dynamics corresponding to the bidding process logic is then

$$s_{i+1,0} = F(s_{i,0}, A_i), \quad \forall i \in \mathtt{I}, \forall A_i \in \mathtt{A}_i \tag{A.5}$$

where the feasible action space $\mathtt{A}_i$ is the set of matrices of actions A_i which satisfy the constraints at time $i \in \mathtt{I}$ defined by Equations (A.3) and (A.4).

We define a reward function $\rho(s_{i,0}, A_i, \mathbf{p}_i)$ for day i which measures the revenues generated by taking a sequence of actions A_i when starting from the state $s_{i,0}$, function of the vector of prices of electricity $\mathbf{p}_i$ for day i. The value of the reward function is given by the total amount of money paid or collected when transferring energy to and from the grid. For every day i, the reward function is defined by

$$\rho(s_{i,0}, A_i, \mathbf{p}_i) = \sum_{t=t_{i,0}}^{t_{i,0}+n-1} r(\mathbf{a}_t, p_t)$$

where $r(\mathbf{a}_t, p_t)$ is given by

$$r(\mathbf{a}_t, p_t) = \left(a_t^{RG}\eta^d - \frac{a_t^{GR}}{\eta^c}\right) p_t$$

with η^d and η^c being the discharge and charge efficiencies, respectively.

In the context of the day-ahead energy market developed in Section A.2, the prices of electricity are known one day before, i.e. the prices of electricity on day $i \in \mathtt{I}$ are known when choosing the sequence of actions $A_i \in \mathtt{A}_i$. An admissible policy $\pi(i, s_{i,0}) : \mathtt{I} \times \mathtt{S} \to \mathtt{A}$ is a function that maps states into actions such that, for any state $s_{i,0}$, the action $\pi(i, s_{i,0})$ satisfies the constraints (A.3) and (A.4) (which defines the set of feasible actions $\mathtt{A}_i \subset \mathtt{A}$). We denote by Π such a set:

$$\Pi = \{\pi : \mathtt{I} \times \mathtt{S} \to \mathtt{A} : \forall s_{i,0} \in \mathcal{S}, \forall i \in \mathtt{I}, \pi(i, s_{i,0}) \in \mathtt{A}_i\}$$

Arguably, the decision of a policy π to be made during the bidding process is whether to buy or sell energy to maximize the revenues on the long term. An optimal value function $V^*_{i+1}(s_{i+1,0})$ is introduced as the maximum expected revenue that can be obtained from time $(i+1,0) = (i, n_H)$ over the remaining time-steps:

$$\forall s_{i+1,0} \in \mathcal{S}, V^*_{i+1}(s_{i+1,0}) = \max_{(A_{i+1},\ldots,A_{n_D-1})\in A_{i+1}\times\ldots\times A_{n_D-1}} \mathop{\mathbb{E}}_{\mathbf{p}_{i+1},\ldots,\mathbf{p}_{n_D-1}} \left[\sum_{k=i+1}^{n_D-1} \rho(s_{k,0}, A_k, \mathbf{p}_k)\right]$$

From these value functions, an optimal policy $\pi^* \in \Pi$ can be defined as follows:

$$\begin{aligned} &\forall i \in \mathtt{I}, \forall s_{i,0} \in \mathcal{S}, \\ &\pi^*(i, s_{i,0}) \in \arg\max_{A_i \in A_i} \left(\rho(s_{i,0}, A_i, \mathbf{p}_i) + V^*_{i+1}(s_{i+1,0})\right) \end{aligned} \tag{A.6}$$

A.4 A DYNAMIC PROGRAMMING APPROACH TO COMPUTE THE OPTIMAL REVENUE OF STORAGE

In this annex, we make the (strong) assumption that the evolution of the prices is perfectly known. This has the two following consequences on the resolution of the above-described problem: (i) the problem becomes deterministic and (ii) the day-ahead structure of the problem disappears.

Let Q_0, Q_1, ..., Q_{24*n_D-1} be the sequence of functions defined as follows:

$$\begin{aligned} &\forall (s, a) \in \mathcal{S} \times \mathcal{A}, t = 0 \ldots n_D * 24 - 1, \\ &Q_t(s, a) = r(s, a, p_t) + \max_{\text{feasible } a' \in \mathcal{A}} Q_{t+1}(f(s, a), a') \end{aligned} \tag{A.7}$$

with

$$Q_{n_D*24}(s, a) = 0, \quad \forall (s, a) \in \mathcal{S} \times \mathcal{A}.$$

It is straightforward to see that when the prices are known we have:

$$V^*_i(s) = \max_{a \in A_i} Q_{i*24}(s, a)$$

From the sequence of functions Q_t, it is possible to estimate in a straightforward way the maximum revenue that can be generated by our storage capacity. We suggest to approximate the computation of this sequence of functions by discretizing the state and the action space Busoniu et al. [2010]; Ernst [2003]. More specifically, the state space is discretized into a set $\{\sigma^{(i)}, i = 1 \ldots n_{\mathcal{S}}\}$, and the action space is discretized into a set $\{\alpha^{(i)}, i = 1 \ldots n_{\mathcal{A}}\}$. We also choose a projection

function $\Gamma : \mathcal{S} \to \{\sigma^{(1)}, \ldots, \sigma^{(n_\mathcal{S})}\}$ which projects any element of the state space $\mathcal{S}$ into a unique element of the discretized space. In such a context, the problem is reduced to a dynamic programming problem with a finite horizon of $n_D * 24$ time-steps that can be solved with a backward value iteration algorithm Powell [2009]. The resulting algorithm is sketched in Procedure 1. It has a complexity proportional to the product of the size of the state space, the action space and the optimization horizon. $\mathfrak{A}(\sigma)$ denotes the set of feasible discretized actions for a given discretized state σ so that the maximization over possible actions $\alpha^{(i)}$ takes into account the constraints stated in Equations (A.3) and (A.4).

From the sequence of $\widehat{Q}_t$ functions outputted by Procedure 1, one can extract a bidding policy. Note that the near-optimal revenue that is obtained from an initial state s_0 can be calculated as follows:

$$\underset{\alpha' \in \mathfrak{A}(\Gamma(s_0))}{\arg\max} \; \widehat{Q}_0(\Gamma(s_0), \alpha')$$

Another way to calculate this revenue is to simulate the system with the policy extracted from these $\widehat{Q}_t$ functions. As way of example, Procedure 2 provides a way for computing the sequence of actions outputted by this policy when the initial state of the system is s_0.

Procedure 1 Q-iteration in the discretized state-action space

Input: $p_t, \; \forall t = 0, ..., n_D * 24 - 1;$

for $t = n_D * 24 - 1$ **to** 0 **do** {Backward loop over all time periods}
  for $\sigma = \sigma^{(1)} \ldots \sigma^{(n_\mathcal{S})}$ **do** {Loop over discretized states}
    for $\alpha = \alpha^{(1)} \ldots \alpha^{(n_\mathcal{A})}$ **do** {Loop over actions}
      $\widehat{Q}_t(\sigma, \alpha) = r(\sigma, \alpha, p_t) + \max_{\alpha' \in \mathfrak{A}(\sigma')} \widehat{Q}_{t+1}(\sigma', \alpha')$ where $\sigma' = \Gamma(f(\sigma, \alpha))$
    end for
  end for
end for
return $\widehat{Q}_t, \forall t \in 0, \ldots, n_D * 24 - 1$

A.5 MATHEMATICAL MODEL FOR ENERGY STORAGE UNDER THE FORM OF HYDROGEN

Each storage capacity is defined by its maximum capacity, its maximum power consumption and restitution to the network as well as the efficiencies for those three steps. A hydrogen-based high-capacity storage device is composed from three main parts: (i) an electrolyzer that transforms water into hydrogen using electricity (ii) a tank where the hydrogen is stored (iii) a fuel cell where the hydrogen is transformed into electricity. Figure A.2 gives a schematic representation of such a device, whose main 3 elements are detailed hereafter.

Procedure 2 Computation of the sequence of actions generated by the bidding policy

Input: $\widehat{Q}_t, \forall t \in 0, 1, \ldots, n_D * 24 - 1$; s_0

$\sigma_0 = \Gamma(s_0)$
for $t = 0$ **to** $n_D * 24 - 1$ **do** {Loop over all time periods}
  $\alpha_t^* = \underset{\alpha' \in \mathfrak{A}(\sigma_t)}{\arg\max}\ \widehat{Q}_t(\sigma', \alpha')$
  $\sigma_{t+1} = f(\sigma_t, \alpha_t^*)$
end for
return $\alpha_t^*, \forall t \in 0, \ldots, n_D * 24 - 1$

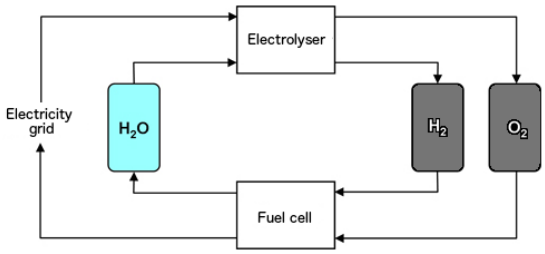


Figure A.2: Sketch of the hydrogen based high-capacity storage.

A.5.1 *Electrolysis*

Currently the dominant technology for direct production of hydrogen (95%) is steam reforming from fossil fuels. However sustainable techniques also exist, such as electrolysis of water using electricity from one of the many renewable sources. It also has the advantage of producing high-purity hydrogen (>99.999%).

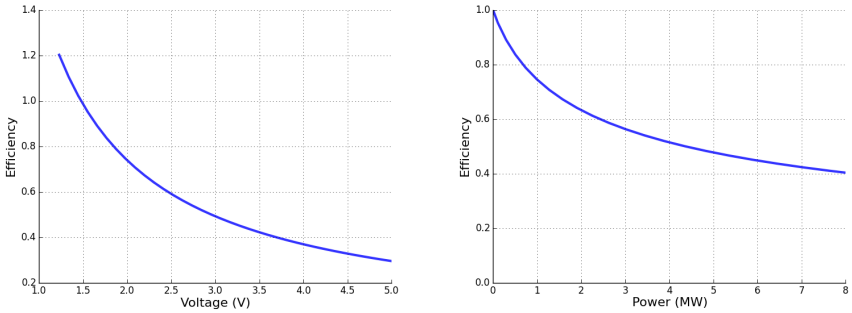
The technical performance of this process has a strong dependency on the rate at which the electrolysis is forced. The charge energy efficiency as a function of the cell voltage is given by:

$$\eta^c = \frac{1.48}{\text{CellVoltage}}$$

The minimum voltage necessary for electrolysis is 1.23 V. Henceforth, the process can theoretically reach efficiencies above 100% but the rate at which the reaction happens is then very low Dopp [2007]. The part of the voltage that exceeds 1.23 V is called overpotential or overvoltage, and leads to losses in the electrochemical process while allowing a higher rate in the reaction. Current density as a function of voltage is approximated at standard temperature for Flat-Plate Bi-functional Cells by

$$I = s \times (\text{CellVoltage} - 1.48),$$

where s is a constant dependent on the setup used for the electrolysis. The evolution of the efficiency with the voltage and with the power generated can be seen on Fig. A.3a and A.3b, respectively.



(a) Efficiency as a function of Voltage (b) Efficiency as a function of Power

Figure A.3: Evolution of the efficiency of the electrolysis process as a function of the rate at which the electrolysis is forced. Parameters used can be found in Table A.1.

A.5.2 *Fuel cell*

A fuel cell is a device that converts the chemical energy from a fuel, in this case hydrogen, into electricity through a chemical reaction with oxygen or another oxidizing agent. Unlike heat engine, the efficiency of a fuel cell is not limited by the Carnot cycle and has a theoretical discharge efficiency $\eta^d = 83\%$ in the case of hydrogen. This efficiency is however lowered when the amount of power generated by the fuel cell increases as illustrated on Fig. A.4. In standard operating conditions, the function $\eta^d(W_{fc})$ can be approximated as a linear equation:

$$\eta^d = \eta^d_{max} - s_{fc} W_{fc}$$

where s_{fc} is a constant dependent on the setup used for the fuel cell and W_{fc} is the power density of the fuel cell.

A.5.3 *The storage device*

One significant constraint that influences the choice of the storage device technology is often the energy density imposed by the application. In the case where hydrogen is to be used as a fuel stored on board of a vehicle, pure hydrogen gas must be pressurized or liquefied. The drawback is that it necessitates the use of external energy to power the compression. This constraint does not hold for grid energy storage, especially in the case where hydrogen can be stored in natural reservoirs such as in underground caverns, salt domes or depleted oil/gas fields.

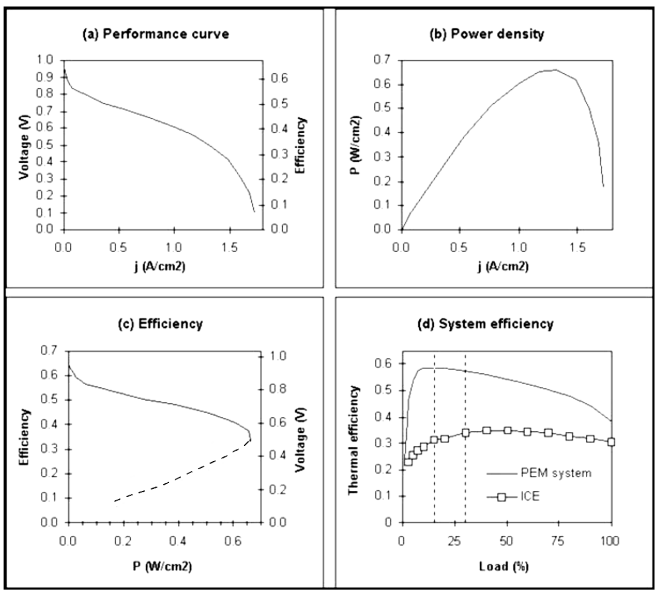


Figure A.4: Characteristics of a PEM fuel cell Kartha and Grimes [2008]. (a) The voltage as a function of the current density, commonly referred to as the performance curve. The efficiency is proportional to the voltage; it is indicated on the secondary vertical axis. (b) The power density as a function of the current density. (c) The efficiency as a function of the power density. The dotted line corresponds to the regime above maximum power. (d) The efficiency of a complete fuel cell system in a vehicle, as a function of power load, shown both for a PEM and an ICE. The vertical dotted lines indicate average loads in a car (left) and a bus or truck (right). The curves in (d) do not refer to the same fuel cell as in (a) to (c).

In the following, the storage device will be characterized by the energy capacity of the device R^c (in MWh). It will be assumed that any leak in the storage device can be neglected.

A.6 EXPERIMENTAL RESULTS

In the first part of this section, the algorithm described in Procedure 1 will be used to figure the maximum revenues that could be generated over the period ranging from 2007 to 2013 by a high-capacity storage device whose parameters are defined in Table A.1. The historical data of electricity prices provided by Belpex over the last few years will be used as input Belpex [2014]. In the second part, the influence of the discretization of the algorithm will be studied. Finally, the impact of the storage capacity on the overall gain will be analyzed.

A.6.1 *Base case*

To compute the revenues of the storage capacity defined by Table 1, we have first discretized the state-action space to be able to use Procedure 1. We choose for the state space a discretized step $\delta_s = 0.5$ MWh. The discretization step for the action space is taken equal to $\delta_u = 0.5$ MWh. That leads to a discretized state space equal to : $\{0, 0.5, 1, \ldots, R_c\}$ and a discretized action space equal to the finite set :$\{-2, -1.5, -1, -0.5, 0, 0.5, 1, 1.5, 2\}$.

Electrolysis	$s_{electrolysis}$	1MA/V
Fuel cell	η^d_{max}	60%
	s_{fc}	0.4 MW^{-1}
	$W_{fc,max}$	5 MW
	$W_{fc,min}$	0.8 MW
Storage device	R^c	1000 MWh
	$s_{(0,0)}$	0 MWh

Table A.1: Data used for the electrolysis sets and fuel cells in the base case.

By using the bidding actions computed using procedures 1 and 2, we have determined the evolution of the cumulative revenues as a function of time. The results are plotted on Fig. A.5. As we can see, the cumulative revenues are not always growing. Indeed, they are decreasing during periods of time when the tank is filled with hydrogen. We note that at the end of the period 2007-2013, a cumulative revenue of 233,000€ is obtained.

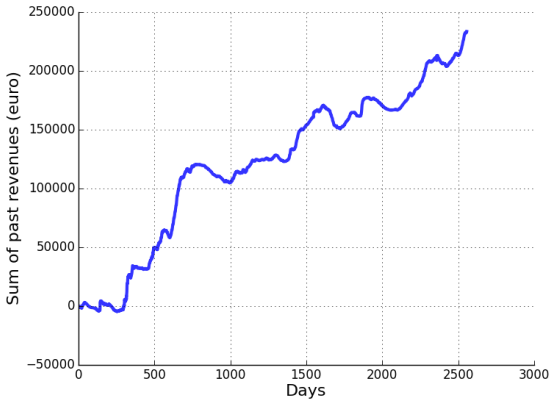
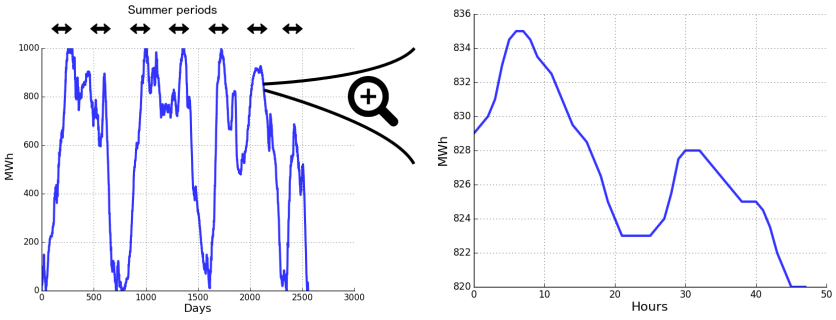


Figure A.5: Cumulative revenues $\sum_t r(\mathbf{a}_t, p_t)$ as a function of time. Day 0 refers to the first of January 2007.

The evolution of the level of energy stored inside the storage tank s_t is shown on Fig. A.6a. It can be seen that hydrogen tends to be stored during summer and transformed back into electricity during winter. This is explained by the fact that most years, prices are higher in

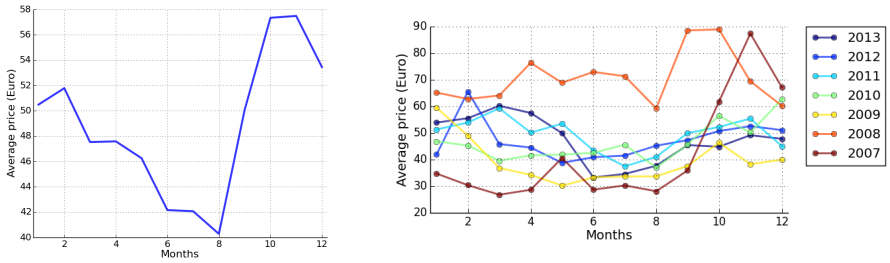
winter and lower in summer (see Fig. A.7). Besides, daily fluctuations can also be seen on Fig A.6b. Energy is accumulated during the night and transferred back to the grid during the day.



(a) Evolution throughout the years 2007 to 2013. Day 0 refers to the first January 2007.

(b) Zoom over two days

Figure A.6: Evolution of the energy reservoir level (s_t) as a function of time for the base case.

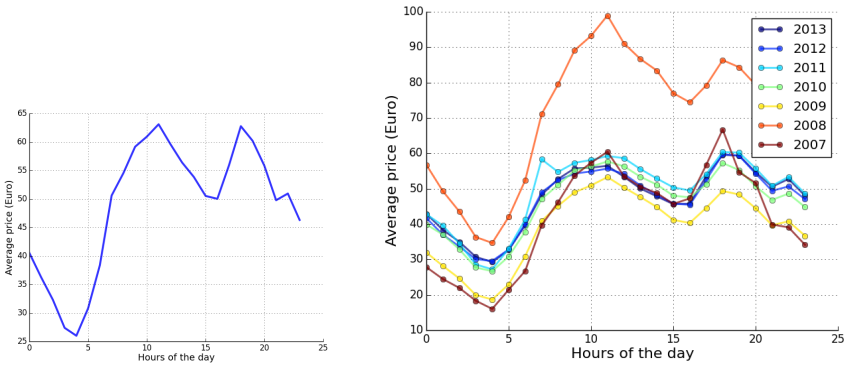


(a) Average over all years

(b) Average over individual years

Figure A.7: Evolution of the average prices for the years 2007 to 2013 as a function of the period of the year.

On Fig. A.8, we have plotted the evolution of the price as a function of the period of the day. We can observe that with the years, the difference between on-peak and off-peak prices tends to decrease. More specifically, the peak prices occurring traditionally during the day tend to get much closer to the average price value. This can be explained by the significant investments that have been made after 2008 in photovoltaic panels. Let us now go back to Fig. A.5 where we have plotted the evolution of the cumulative revenues over time. As one can observe, the rate of growth in cumulative revenues is higher for the first two years than for the rest of the period. This observation is a direct consequence from this flattening of the price evolution over the day.



(a) Average over all years (b) Average over individual years

Figure A.8: Evolution of the price as a function of the hour of the day for the years 2007 to 2013.

Finally, we end this subsection by Fig. A.9, which nicely illustrates on a single graphic the relation that exists behind the evolution of the prices and the sequence of actions taken.

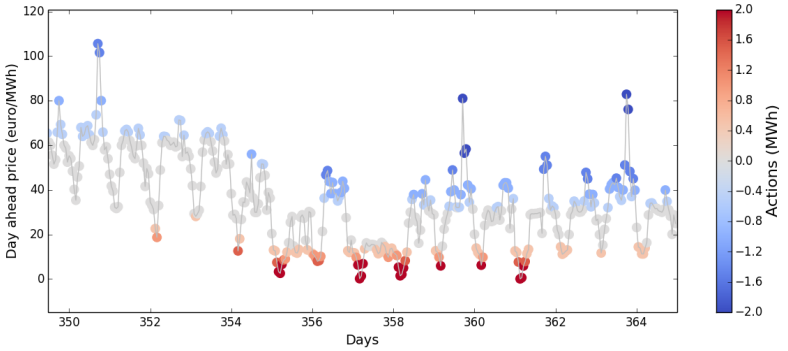


Figure A.9: Illustration of the sequence of actions and of electricity prices with time. A dot is associated to every market period and its color refers to the action taken. Its position gives the market price at this time.

A.6.2 *Influence of the capacity of the storage tank on the maximum revenue*

In this section, we study the revenues obtained as a function of the size of the reservoir. We have modeled the storage reservoir as varying between a few MWh up to a reservoir which is large enough for never being fully filled by the agent. The results are plotted on Fig. A.10. We remind the reader that in the previous subsection, a maximum capacity of 1000 MWh was used for the storage device. As we can see, the revenues are a growing function of the storage capacity. However, the incremental revenue obtained from the exploitation is lowered as the storage capacity increases. Whatever

the size of the reservoir, it is not possible to generate a revenue which is larger than 272 000 €.

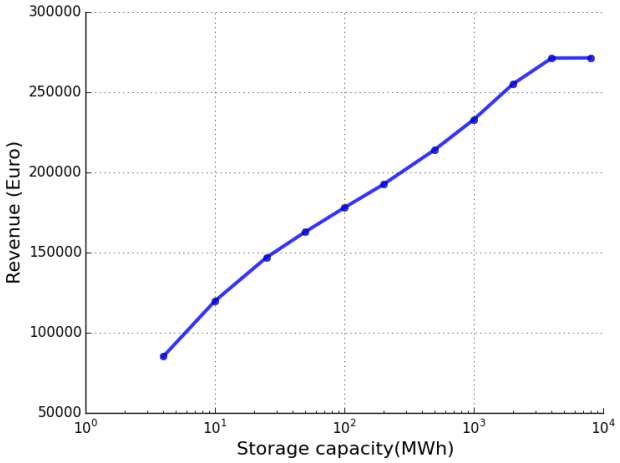


Figure A.10: Evolution of the expected revenues as a function of storage capacity for the years 2007 to 2013.

A.6.3 *Influence of the discretization on the maximum revenue*

In this section, we study the influence of the discretization steps δ_s and δ_u on the results obtained. To do so, we have run Procedure 1, followed by Procedure 2, for several values of δ_s and δ_u. Figure A.11 plots the results obtained. Several interesting observations can be made. First, for a given value of δ_s (δ_u), the return of the bidding policy does not vary anymore when δ_u (δ_s) becomes lower than δ_s (δ_u). Second, if $\delta_u > \delta_s$ ($\delta_s > \delta_u$), better results can be obtained by moving δ_u closer to δ_s (δ_s closer to δ_u). Finally, in the case where the discretization steps are equal, the smaller they are, the better the quality of the policy. Note however, that below a certain value of the discretization steps, the quality of the policy remains roughly the same.

A.7 REVENUES ESTIMATION UNDER MULTIPLE PRICE EVOLUTIONS

The experimental design exposed in the previous sections assumes that the future price evolution is known in advance. Additionally, it also relies on a discretization of the state-action space. A more realistic assumption would be to assume a set of possible price evolutions $\{(p_{k,t})_t\}_{k=1}^{K}$ where $K \in \mathbb{N}_0$ and consider the empirical return of a given sequence of actions over such a set of realizations as an approximation of the expected value of this sequence of actions.

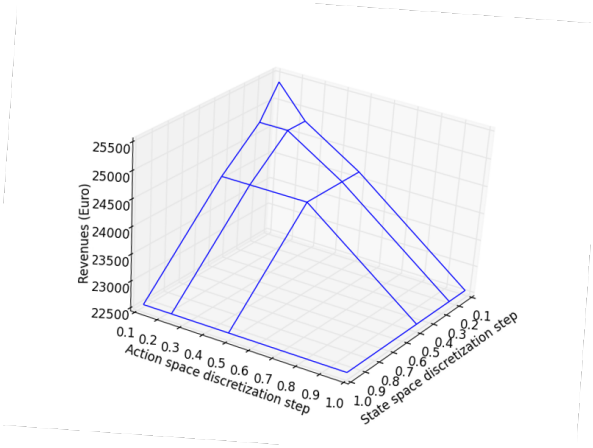


Figure A.11: Revenues generated by the bidding policies as a function of the action space discretization step (δ_u) and the state space discretization step (δ_s).

If we make the hypothesis that the efficiencies of the fuel cell and the electrolyzer are constant, the reward function and the system dynamics are both linear mappings of states, actions and prices. The empirical return over K price outcomes is then also a linear mapping of the control variables. Finding a sequence of actions $\left(A_0^*, \ldots, A_{n_D-1}^*\right)$ leading to the maximization of the empirical return can be achieved by solving the following linear program:

$$\left(A_0^*, \ldots, A_{n_D-1}^*\right) \in \underset{A_i \in A_i, i=0\ldots n_D-1}{\arg\max} \frac{1}{K} \sum_{k=1}^{K} \sum_{i=0}^{n_D-1} \rho(s_{i,0}, A_i, \mathbf{p}_{k,t}),$$

where $\mathbf{p}_{k,t}$ denotes the vector of prices for the $i-$th day. Note that the previous linear program allows to solve the problem exposed in the previous section in the case $K = 1$ for continuous state-action spaces. The dynamic programming approach has the advantage (i) to ensure a linear complexity with the time horizon and (ii) to remain applicable in the case of non-linear and even non-convex system dynamics and reward function.

A.8 CONCLUSION

In this annex, a methodology has been proposed for estimating the revenues that can be generated by a high-capacity hydrogen-based storage device on the energy markets. It was then used to estimate the revenues that could be generated on Belpex - the Belgian power exchange market.

The results show that for fixed size electrolyzers and fuel cells, significantly higher revenues can be achieved by having large storage capacities, such as for example hydrogen tanks that would take tens

of days to fill or to empty. This is explained by the fact that with huge tanks, the storage device can be operated so as to exploit the inter-seasonal price fluctuations. The results also show that over the last years, the revenues that could have been generated by storage devices have decreased.

The research reported in this annex could be extended along several directions. First, our algorithm for estimating the future revenues assumes that the market price is not influenced by the storage device itself and, more importantly, that the future price evolution is known (or, at least, an ensemble of possible price evolutions are known). It would be worth extending the methodology proposed in this annex to a more general case. Note that this would imply working in a probabilistic setting where we would compute an expected future revenue or a distribution over future revenues.

Second, the only mechanism considered here for valorizing storage has been to buy or sell energy on the electricity market. But other mechanisms also exist, such as for example selling services to the balancing/reserves markets Eurelectric [2012] or those that would relate to absorbing the excess of energy produced locally by renewable sources of energy so as to relieve congestions Gemine et al. [2013]. In this respect, it would be worth computing the revenues that can be generated by storage devices when all these mechanisms are taken into account.

Finally, it would be interesting to study how prediction models of the future revenues could be utilized to give clear indications about the storage technology in which to invest and about where to install storage devices.

B

NOTATIONS

NOTATIONS INTRODUCED IN INTRODUCTION

$\mathcal{X}$	input space of a function	10
x	input of a function	10
$\mathcal{Y}$	output space of a function	10
y	output of a function	10
n_θ	number of parameters in a neural network	10
$f(x;\theta)$	function of x, parametrized by θ	10
θ	parameters of a function	10
$\mathbb{N}$	positive integers	10
h	hidden layer values	10
W_1, W_2	weight matrix	10
b_1, b_2	weight vector (bias)	10
ω_t	observation obtained at time t	15
Ω	observation space	15
t	timestep	15
s_t	state of the agent within its environment at time t	16
s	state of the agent within its environment	16
$\mathcal{S}$	state space of a MDP	16
a_t	action taken at time t	15
a	action	15
$\mathcal{A}$	action space	15
r_t	reward obtained at time t	16
$\mathcal{R}$	reward space	16
$R(\cdot,\cdot,\cdot)$	reward function	16
$T(\cdot,\cdot,\cdot)$	transition function	16
$N_\mathcal{S}$	Number of discrete states	16
$N_\mathcal{A}$	Number of discrete actions	16
R_{max}	constant representing the maximum possible reward	16
$\pi(\cdot)$	determinisitc policy	17
$\pi(\cdot,\cdot)$	stochastic policy	17
Π	policy space	17
$V^\pi(\cdot)$	value function conditioned on policy π	17
$V^*(\cdot)$	optimal value function	18
α_k	weighting factors in the expected return	17

γ	discount factor	17
H	horizon	17
$T_{(s,a)}$	discrete random variable of transitions frequencies	18
$R_{(s,a)}$	continuous random variable of rewards	19
D_s	dataset obtained in a MDP	19
$\mathcal{D}_s$	distribution of datasets obtained in a MDP	19
$Q^\pi(\cdot,\cdot)$	action-value function conditioned on policy π	20
$Q^*(\cdot,\cdot)$	optimal action-value function	20
$\pi^*(\cdot)$	optimal policy	20
$\mathcal{B}$	Bellman operator	21
$Q(\cdot,\cdot;\theta)$	Q-network (neural network aiming at fitting an action-value function	22
Y_k^Q	target value for a Q-network	22
θ_k	parameters of a Q-network at the k^{th} iteration	22
θ_k^-	parameters of the target Q-network at the k^{th} iteration	22
s'	next state of the agent (following s)	22
α	learning rate	22
N_{replay}	size (in number of tuples) of the replay memory	23
$A^\pi(\cdot,\cdot)$	advantage function	23
$R'(\cdot,\cdot)$	reward function	24
$\rho^\pi(s)$	discounted state distribution	24
$\pi_\theta(\cdot)$	differentiable deterministic policy	25
$\pi_\theta(\cdot,\cdot)$	differentiable stochastic policy	25
$\pi_k(\cdot)$	deterministic policy at the k^{th} iteration	26

NOTATIONS INTRODUCED IN PART I

N_Ω	Number of discrete observations	44
$O(\cdot\|\cdot)$	conditional observation probabilities	44
$b(\cdot)$	distribution of states	44
$\mathbb{N}_0$	non-negative integers	45
$\mathcal{H}_t$	set of histories observed up to time t	45
$\mathcal{H}$	space of all possible observable histories	45
$\phi(\cdot)$	mapping	46
$b(\cdot\|\cdot)$	belief state	46
H_t	history of observations up to time t	46

$\phi_0(\cdot)$	mapping providing sufficient statistics .. 46
$\phi_\epsilon(\cdot)$	mapping providing ϵ-sufficient statistics 46
ϵ	arbitrarily small real number ($\geqslant 0$) 46
$\mathcal{M}(\mathcal{S}, \mathcal{A}, \Omega, \gamma)$	set of POMDPs 46
$M(T, R, O)$ or M	POMDP 46
D_{M,π_s,N_{tr},N_l} or D	dataset 46
$\mathcal{D}_{M,\pi_s,N_{tr},N_l}$	probability distribution of datasets 46
π_s	stochastic sampling policy 46
N_{tr}	number of trajectories 46
N_l	number of timesteps of a trajectory 46
$\hat{M}_{D,\phi}$ or $\hat{M}_D$	frequentist-based MDP 47
$V^\pi_M(\cdot)$ or $\mathcal{V}^\pi_{\hat{M}_D}(\cdot)$	value function conditioned on policy π in environment $M/\hat{M}_D$ 47
Σ	state space of a frequentist-based MDP .. 47
σ	state of a frequentist-based MDP 47
A	action space of a frequentist-based MDP 47
$\hat{T}(\cdot,\cdot,\cdot)$	estimated transition function of a frequentist-based MDP 47
$\hat{R}(\cdot,\cdot,\cdot)$	estimated reward function of a frequentist-based MDP 48
Γ	frequentist-based MDPs discount factor . 48
$R_{model-based}(\cdot,\cdot)$	model based reward function 48
$T_{model-based}(\cdot,\cdot,\cdot)$	model based transition function 48
$\hat{r}_t$	reward in the frequentist-based MDP 48
$\pi_{D,\phi}(\cdot)$	deterministic frequentist-based policy ... 48
$\mathcal{F}_{M,\phi}$	function class 52
$f^\pi_{M,\phi}$	function 52
$D_{\varphi,a}$	set of n pairs of immediate reward and next-state representation sampled from (φ, a) in dataset D 52
$\hat{\mathfrak{R}}_{D_{\varphi,a}}(\mathcal{F}_{M,\phi})$	empirical Rademacher complexity of the function class $\mathcal{F}_{M,\phi}$ 52
N_P	number of POMDPs sampled from a distribution 53
$\mathcal{P}$	distribution of random POMDPs 53
$\mathcal{D}_P$	distribution of random datasets 53
μ_P	average score 53
σ_P	parametric standard deviation 54
$\hat{R}'(\cdot,\cdot)$	estimated reward function of a frequentist-based MDP 58

w_t	random disturbance	66
$\mathcal{W}$	disturbance space	66
γ_k	discount factor at the k^{th} epoch	66

NOTATIONS INTRODUCED IN PART II

$\mathcal{T}$	set of time periods	77
Δt	Duration of one time step	77
J	number of different photovoltaic technologies	77
L	number of different battery technologies	77
M	number of different hydrogen technologies	77
$\mathcal{E}$	environment space	77
$\mathbf{E}_t$	time-varying environment vector	77
c_t	electricity demand	77
i_t	solar irradiance	77
$\mathbf{e}^{PV}_{j,t}$	photovoltaic model	77
$\mathcal{E}^{PV}_j$	ensemble of possible photovoltaic models	77
$c^{PV}_{j,t}$, c^{PV}_t, c^{PV}	photovoltaic cost	77
$L^{PV}_{j,t}$, L^{PV}_t, L^{PV}	photovoltaic lifetime	77
$\eta^{PV}_{j,t}$, η^{PV}_t, η^{PV}	photovoltaic efficiency	77
$\mathbf{e}^{B}_{l,t}$	battery model	77
$\mathcal{E}^{B}_l$	ensemble of possible battery models	77
$c^{B}_{j,t}$, c^{B}_t, c^{B}	battery cost	77
$L^{B}_{j,t}$, L^{B}_t, L^{B}	battery lifetime	77
$D^{B}_{l,t}$, D^{B}_t, D^{B}	battery cycle durability	77
$P^{B}_{l,t}$, P^{B}_t, P^{B} [W]	battery power limit for charge and discharge	77
$\eta^{B}_{l,t}$, η^{B}_t, η^{B}	battery charge efficiency	77
$\zeta^{B}_{l,t}$, ζ^{B}_t, ζ^{B}	battery discharge efficiency	77
$r^{B}_{l,t}$, r^{B}_t, r^{B}	battery charge retention rate	77
$\mathbf{e}^{H_2}_{m,t}$	hydrogen model	77
$\mathcal{E}^{H_2}_m$	ensemble of possible hydrogen models	77
$c^{H_2}_{m,t}$, $c^{H_2}_t$, c^{H_2}	hydrogen cost	77
$L^{H_2}_{m,t}$, $L^{H_2}_t$, L^{H_2}	hydrogen lifetime	77
$R^{H_2}_{m,t}$, $R^{H_2}_t$, R^{H_2}	hydrogen maximum capacity	77

$\eta^{H_2}_{m,t}$, $\eta^{H_2}_t$, η^{H_2}	hydrogen charge efficiency 77
$\zeta^{H_2}_{m,t}$, $\zeta^{H_2}_t$, ζ^{H_2}	hydrogen discharge efficiency 78
$r^{H_2}_{m,t}$, $r^{H_2}_t$, r^{H_2}	hydrogen charge retention rate 78
$\boldsymbol{\mu}_t$	model of interaction 78
$\mathcal{J}$	ensemble of models of interaction 78
β	price at which it is possible to sell energy to the grid 78
k	cost endured per kWh that is not supplied within the microgrid 78
$\mathbf{s}_t$	time varying vector characterizing the microgrid's state 78
$\mathbf{s}^{(s)}_t$	state information related to the infrastructure . 78
$\mathcal{S}^{(s)}$	state space related to the infrastructure 78
$\mathbf{s}^{(o)}_t$	state information related to the operation 78
$\mathcal{S}^{(o)}$	state space related to the operation 78
x^{PV}_t, x^{PV}	sizing of the PV panels 79
x^{B}_t, x^{B}	sizing of the battery 79
$x^{H_2}_t$, x^{H_2}	sizing of the hydrogen storage 79
s^{B}_t	level of charge of the battery 79
$s^{H_2}_t$	level of charge of the hydrogen storage 79
$\mathbf{a}$	action vector in the microgrid environment ... 79
$\mathbf{a}^{(s)}_t$	sizing actions 79
$\mathcal{A}^{(s)}$	sizing actions space 79
$\mathbf{a}^{(o)}_t$	control actions 79
$\mathcal{A}^{(o)}_t$	control actions space 79
a^{PV}_t	new sizing of the PV panels 80
a^{B}_t	new sizing of the battery 80
$a^{H_2}_t$	new sizing of the hydrogen device 80
p^{B}_t [W]	power provided to the battery 80
$p^{H_2}_t$	power provided to the hydrogen storage 80
$\mathcal{G}_T$	set of positive scalar functions 81
G_T	positive scalar function 81
$\mathcal{G}_0$	set of positive scalar functions 82
G_0	positive scalar function 82
n	lifetime of the microgrid in years 84
I_0	initial investment expenditures 84
M_y	operational expenses in the year y 84
ϵ_y	electricity consumption in the year y 84
ρ	discount rate which may refer to the interest rate or to the discounted cash flow 84

τ_y	set of time steps belonging to year y 84
r_t	revenues obtained at each time step t 84
ψ_t	electricity generated by the photovoltaic installation ... 85
d_t	net electricity demand 85
δ_t	power balance within the microgrid 85
$\mathcal{M}_{op}$	linear program for the optimal control 87
$\mathcal{M}_{size}$	linear program for the optimal sizing and operation ... 88
$\mathcal{M}_{rob}$	linear program for the robust optimization of the sizing under optimal operation 88
r^{H_2}	revenues generated by the hydrogen production 100
r^{-}	penalties due to the value of loss load 100
$\widetilde{Q}^*$	best (approximated) Q-network 102
h^c	number of hours of a consumption time series 103
h^p	number of hours of a production time series 103
ζ_s	smallest number of days to the solar solstice 103
ρ_n	solar production for the next n hours 103
ρ_{RMS}	parameter in the RMSprop algorithm (ρ close to 1 corresponds to a slow decay of the moving average) 110
ϵ	percentage of random action in an ϵ-greedy policy ... 110

C

REFERENCES

P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1. ACM, 2004.

D. Abel, D. Hershkowitz, and M. Littman. Near optimal behavior via approximate state abstraction. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 2915–2923, 2016.

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.

C. Archer and M. Jacobson. Supplying baseload power and reducing transmission requirements by interconnecting wind farms. *Journal of Applied Meteorology & Climatology*, 46(11), 2007.

N. Armaroli and V. Balzani. The hydrogen issue. *ChemSusChem*, 4(1): 21–36, 2011.

P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

P.-L. Bacon, J. Harb, and D. Precup. The option-critic architecture. *arXiv preprint arXiv:1609.05140*, 2016.

D. Bahdanau, P. Brakel, K. Xu, A. Goyal, R. Lowe, J. Pineau, A. Courville, and Y. Bengio. An actor-critic algorithm for sequence prediction. *arXiv preprint arXiv:1607.07086*, 2016.

L. Baird. Residual algorithms: Reinforcement learning with function approximation. In *ICML*, pages 30–37, 1995.

P. L. Bartlett and S. Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002.

M. Bazilian, I. Onyeji, M. Liebreich, I. MacGill, J. Chase, J. Shah, D. Gielen, D. Arent, D. Landfear, and S. Zhengrong. Re-considering the economics of photovoltaic power. *Renewable Energy*, 53:329–338, 2013.

C. Beattie, J. Z. Leibo, D. Teplyashin, T. Ward, M. Wainwright, H. Küttler, A. Lefrancq, S. Green, V. Valdés, A. Sadik, et al. Deepmind lab. *arXiv preprint arXiv:1612.03801*, 2016.

M. G. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos. Unifying count-based exploration and intrinsic motivation. *arXiv preprint arXiv:1606.01868*, 2016.

R. E. Bellman and S. E. Dreyfus. Applied dynamic programming. 1962.

I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2016.

Belpex, feb 2014. URL http://www.belpex.be.

A. Ben-Tal and A. Nemirovski. Robust optimization–methodology and applications. *Mathematical Programming*, 92(3):453–480, 2002.

Y. Bengio, D.-H. Lee, J. Bornschein, T. Mesnard, and Z. Lin. Towards biologically plausible deep learning. *arXiv preprint arXiv:1502.04156*, 2015.

D. Berthelot, T. Schumm, and L. Metz. Began: Boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*, 2017.

M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

A. Boularias, J. Kober, and J. Peters. Relative entropy inverse reinforcement learning. In *AISTATS*, pages 182–189, 2011.

R. I. Brafman and M. Tennenholtz. R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *The Journal of Machine Learning Research*, 3:213–231, 2003.

D. Braziunas. POMDP solution methods. *University of Toronto, Tech. Rep*, 2003.

G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym, 2016.

N. Brown and T. Sandholm. Libratus: The superhuman ai for no-limit poker. *International Joint Conference on Artificial Intelligence (IJCAI-17)*, 2017.

C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

B. Brügmann. Monte carlo go. Technical report, Citeseer, 1993.

L. Busoniu, R. Babuska, B. De Schutter, and D. Ernst. *Reinforcement learning and dynamic programming using function approximators*. CRC Press, 2010.

R. Carmona and M. Ludkovski. Valuation of energy storage: An optimal switching approach. *Quantitative Finance*, 10(4):359–374, 2010.

M. Castronovo, V. François-Lavet, R. Fonteneau, D. Ernst, and A. Couëtoux. Approximate bayes optimal policy search using neural networks. In *9th International Conference on Agents and Artificial Intelligence (ICAART 2017)*, 2017.

S. Chatzivasileiadis, D. Ernst, and G. Andersson. The global grid. *Renewable Energy*, 57:372–383, 2013.

S. Chatzivasileiadis, D. Ernst, and G. Andersson. Global power grids for harnessing world renewable energy. *To be published by Elsevier*, 2014.

T. Chen, I. Goodfellow, and J. Shlens. Net2net: Accelerating learning via knowledge transfer. *arXiv preprint arXiv:1511.05641*, 2015.

X. Chen, C. Liu, and D. Song. Learning neural programs to parse programs. *arXiv preprint arXiv:1706.01284*, 2017.

H. A. Chih-Chiang and S. B. Zong-Wei. Charge and discharge characteristics of lead-acid battery and lifepo4 battery. In *Power Electronics Conference (IPEC), 2010 International*, pages 1478–1483. IEEE, 2010.

D. Connolly, H. Lund, B. V. Mathiesen, and M. Leahy. The first step towards a 100% renewable energy-system for Ireland. *Applied Energy*, 88(2):502–507, 2011.

CREG. Study (f)121011-cdc-1182 on "capacity remuneration mechanisms". Technical report, CREG, 2012.

Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai. Deep direct reinforcement learning for financial signal representation and trading. *IEEE transactions on neural networks and learning systems*, 28(3):653–664, 2017.

R. Dopp. Hydrogen generation via water electrolysis using highly efficient nanometal electrodes. *Quantum Sphere, Inc*, 2007.

Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel. Rlˆ2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

L. Duchesne, E. Karangelos, and L. Wehenkel. Machine learning of real-time power systems reliability management response. *PowerTech Manchester 2017 Proceedings*, 2017.

S. Džeroski, L. De Raedt, and K. Driessens. Relational reinforcement learning. *Machine learning*, 43(1-2):7–52, 2001.

D. Ernst. *Near optimal closed-loop control Application to electric power systems*. PhD thesis, University of Liege, 2003.

D. Ernst, P. Geurts, L. Wehenkel, and M. L. Littman. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6(4), 2005.

Eurelectric. Decentralized storage: Impact on future distribution grids. Technical report, Eurelectric, 2012.

J. Eyer and G. Corey. Energy storage for the electricity grid: Benefits and market potential assessment guide. *Sandia National Laboratories*, 2010.

C. Fernando, D. Banarse, M. Reynolds, F. Besse, D. Pfau, M. Jaderberg, M. Lanctot, and D. Wierstra. Convolution by evolution: Differentiable pattern producing networks. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference*, pages 109–116. ACM, 2016.

N. Ferns, P. Panangaden, and D. Precup. Metrics for finite markov decision processes. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 162–169. AUAI Press, 2004.

H. L. Ferreira, R. Garde, G. Fulli, W. Kling, and J. P. Lopes. Characterisation of electrical energy storage technologies. *Energy*, 53:288–298, 2013.

C. Finn, I. Goodfellow, and S. Levine. Unsupervised learning for physical interaction through video prediction. In *Advances In Neural Information Processing Systems*, pages 64–72, 2016a.

C. Finn, S. Levine, and P. Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48, 2016b.

S.-E. Fleten and T. K. Kristoffersen. Stochastic programming for optimizing bidding strategies of a nordic hydropower producer. *European Journal of Operational Research*, 181(2):916–928, 2007.

C. Florensa, Y. Duan, and P. Abbeel. Stochastic neural networks for hierarchical reinforcement learning. *arXiv preprint arXiv:1704.03012*, 2017.

J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson. Counterfactual multi-agent policy gradients. *arXiv preprint arXiv:1705.08926*, 2017.

D. B. Fogel. *Evolutionary computation: toward a new philosophy of machine intelligence*, volume 1. John Wiley & Sons, 2006.

R. Fonteneau, L. Wehenkel, and D. Ernst. Variable selection for dynamic treatment regimes: a reinforcement learning approach. 2008.

R. Fonteneau, S. A. Murphy, L. Wehenkel, and D. Ernst. Model-free monte carlo-like policy evaluation. In *AISTATS*, pages 217–224, 2010.

V. François-Lavet, R. Fonteneau, and D. Ernst. Using approximate dynamic programming for estimating the revenues of a hydrogen-based high-capacity storage device. In *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL), 2014*, pages 1–8. IEEE, 2014.

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, 2016a.

V. François-Lavet, D. Taralla, D. Ernst, and R. Fonteneau. Deep reinforcement learning solutions for energy microgrids management. In *European Workshop on Reinforcement Learning*, 2016b.

K. Fukushima and S. Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer, 1982.

D. Gandhi, L. Pinto, and A. Gupta. Learning to fly by crashing. *arXiv preprint arXiv:1704.05588*, 2017.

J. Garcıa and F. Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1): 1437–1480, 2015.

M. Garnelo, K. Arulkumaran, and M. Shanahan. Towards deep symbolic reinforcement learning. *arXiv preprint arXiv:1609.05518*, 2016.

S. Gelly, Y. Wang, R. Munos, and O. Teytaud. Modification of UCT with patterns in Monte-Carlo go. 2006.

Q. Gemine, E. Karangelos, D. Ernst, and B. Cornélusse. Active network management: planning under uncertainty for exploiting load modulation. In *Bulk Power System Dynamics and Control-IX Optimization, Security and Control of the Emerging Power Grid (IREP), 2013 IREP Symposium*, pages 1–9. IEEE, 2013.

A. Geramifard, C. Dann, R. H. Klein, W. Dabney, and J. P. How. Rlpy: A value-function-based reinforcement learning framework for education and research. *Journal of Machine Learning Research*, 16:1573–1578, 2015.

M. Ghavamzadeh, S. Mannor, J. Pineau, A. Tamar, et al. Bayesian reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, 8(5-6):359–483, 2015.

A. Giusti, J. Guzzi, D. C. Cireşan, F.-L. He, J. P. Rodríguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. Di Caro, et al. A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robotics and Automation Letters*, 1(2):661–667, 2016.

M. Glavic, R. Fonteneau, and D. Ernst. Reinforcement learning for electric power system decision and control: Past considerations and perspectives. In *The 20th World Congress of the International Federation of Automatic Control, Toulouse 9-14 July 2017*, pages 1–10, 2017.

X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Aistats*, volume 9, pages 249–256, 2010.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT Press, 2016.

G. J. Gordon. Approximate solutions to markov decision processes. *Robotics Institute*, page 228, 1999.

A. Graves, G. Wayne, and I. Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

L. Grégoire, F.-L. Vincent, E. Damien, J. M. Christoph, and S. L. Klaus. Electricity storage with liquid fuels in a zone powered by 100% variable renewables. In *European Energy Market (EEM), 2015 12th International Conference on the European Energy Market (EEM)*, pages 1–5. IEEE, 2015.

A. Gruslys, M. G. Azar, M. G. Bellemare, and R. Munos. The reactor: A sample-efficient actor-critic architecture. *arXiv preprint arXiv:1704.04651*, 2017.

S. Gu, T. Lillicrap, I. Sutskever, and S. Levine. Continuous deep q-learning with model-based acceleration. *arXiv preprint arXiv:1603.00748*, 2016.

H. V. Hasselt. Double q-learning. In *Advances in Neural Information Processing Systems*, pages 2613–2621, 2010.

M. Hausknecht and P. Stone. Deep recurrent q-learning for partially observable MDPs. *arXiv preprint arXiv:1507.06527*, 2015.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

J. Ho and S. Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, pages 4565–4573, 2016.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

S. Hochreiter, A. S. Younger, and P. R. Conwell. Learning to learn using gradient descent. In *International Conference on Artificial Neural Networks*, pages 87–94. Springer, 2001.

W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301):13–30, 1963.

R. Houthooft, X. Chen, Y. Duan, J. Schulman, F. De Turck, and P. Abbeel. Curiosity-driven exploration in deep reinforcement learning via bayesian neural networks. *arXiv preprint arXiv:1605.09674*, 2016.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

N. Jiang and L. Li. Doubly robust off-policy value evaluation for reinforcement learning. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 652–661, 2016.

N. Jiang, A. Kulesza, and S. Singh. Abstraction selection in model-based reinforcement learning. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 179–188, 2015a.

N. Jiang, A. Kulesza, S. Singh, and R. Lewis. The dependence of effective planning horizon on model accuracy. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 1181–1189. International Foundation for Autonomous Agents and Multiagent Systems, 2015b.

J. Johnson, B. Hariharan, L. van der Maaten, J. Hoffman, L. Fei-Fei, C. L. Zitnick, and R. Girshick. Inferring and executing programs for visual reasoning. *arXiv preprint arXiv:1705.03633*, 2017.

S. Kakade, M. Kearns, and J. Langford. Exploration in metric state spaces. In *ICML*, volume 3, pages 306–312, 2003.

M. Kalakrishnan, P. Pastor, L. Righetti, and S. Schaal. Learning objective functions for manipulation. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1331–1336. IEEE, 2013.

N. Kalchbrenner, A. v. d. Oord, K. Simonyan, I. Danihelka, O. Vinyals, A. Graves, and K. Kavukcuoglu. Video pixel networks. *arXiv preprint arXiv:1610.00527*, 2016.

R. Kaplan, C. Sauer, and A. Sosa. Beating atari with natural language guided reinforcement learning. *arXiv preprint arXiv:1704.05539*, 2017.

S. Kartha and P. Grimes. Fuel cells: Energy conversion for the next century. *Physics Today*, 47(11):54–61, 2008.

M. Kearns and S. Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2-3):209–232, 2002.

J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *arXiv preprint arXiv:1612.00796*, 2016.

G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter. Self-normalizing neural networks. *arXiv preprint arXiv:1706.02515*, 2017.

L. Kocsis and C. Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.

G. Krajačić, N. Duić, and M. da Graça Carvalho. H_2 RES, energy planning tool for island energy systems–the case of the island of mljet. *International journal of hydrogen energy*, 34(16):7015–7026, 2009.

V. Kuleshov and D. Precup. Algorithms for multi-armed bandit problems. *arXiv preprint arXiv:1402.6028*, 2014.

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

Y. LeCun, Y. Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361 (10):1995, 1995.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553): 436–444, 2015.

S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17 (39):1–40, 2016.

L. Li, Y. Lv, and F.-Y. Wang. Traffic signal timing via deep reinforcement learning. *IEEE/CAA Journal of Automatica Sinica*, 3(3):247–254, 2016.

X. Li, L. Li, J. Gao, X. He, J. Chen, L. Deng, and J. He. Recurrent reinforcement learning: a hybrid approach. *arXiv preprint arXiv:1509.03044*, 2015.

T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

L.-J. Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3-4):293–321, 1992.

Y. Liu, A. Gupta, P. Abbeel, and S. Levine. Imitation from observation: Learning to imitate behaviors from raw video via context translation. *arXiv preprint arXiv:1707.03374*, 2017.

N. Löhndorf and S. Minner. Optimal day-ahead trading and storage of renewable energies - an approximate dynamic programming approach. *Energy Systems*, 1(1):61–77, 2010.

O.-A. Maillard, D. Ryabko, and R. Munos. Selecting the state-representation in reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2627–2635, 2011.

T. Mandel, Y.-E. Liu, S. Levine, E. Brunskill, and Z. Popovic. Offline policy evaluation across representations with applications to educational games. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 1077–1084. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

S. Mannor, D. Simester, P. Sun, and J. N. Tsitsiklis. Bias and variance approximation in value function estimates. *Management Science*, 53 (2):308–322, 2007.

M. Mathieu, C. Couprie, and Y. LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*, 2015.

A. K. McCallum. *Reinforcement learning with selective perception and hidden state*. PhD thesis, University of Rochester, 1996.

R. Miikkulainen, J. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, A. Navruzyan, N. Duffy, and B. Hodjat. Evolving deep neural networks. *arXiv preprint arXiv:1703.00548*, 2017.

W. Mischel, E. B. Ebbesen, and A. Raskoff Zeiss. Cognitive and attentional mechanisms in delay of gratification. *Journal of personality and social psychology*, 21(2):204, 1972.

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

V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, 2016.

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.

P. Mokrian and M. Stephen. A stochastic programming framework for the valuation of electricity storage. In *26th USAEE/IAEE North American Conference*, pages 24–27, 2006.

M. Moravčík, M. Schmid, N. Burch, V. Lisỳ, D. Morrill, N. Bard, T. Davis, K. Waugh, M. Johanson, and M. Bowling. Deepstack: Expert-level artificial intelligence in no-limit poker. *arXiv preprint arXiv:1701.01724*, 2017.

R. Munos, T. Stepleton, A. Harutyunyan, and M. Bellemare. Safe and efficient off-policy reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 1046–1054, 2016.

A. Neelakantan, Q. V. Le, and I. Sutskever. Neural programmer: Inducing latent programs with gradient descent. *arXiv preprint arXiv:1511.04834*, 2015.

G. Neu and C. Szepesvári. Apprenticeship learning using inverse reinforcement learning and gradient methods. *arXiv preprint arXiv:1206.5264*, 2012.

A. Y. Ng, D. Harada, and S. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, pages 278–287, 1999.

A. Y. Ng, S. J. Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, pages 663–670, 2000.

D. H. Nguyen and B. Widrow. Neural networks for self-learning control systems. *IEEE Control systems magazine*, 10(3):18–23, 1990.

B. O'Donoghue, R. Munos, K. Kavukcuoglu, and V. Mnih. Pgq: Combining policy gradient and q-learning. *arXiv preprint arXiv:1611.01626*, 2016.

J. Oh, X. Guo, H. Lee, R. L. Lewis, and S. Singh. Action-conditional video prediction using deep networks in atari games. In *Advances in Neural Information Processing Systems*, pages 2863–2871, 2015.

J. Oh, S. Singh, and H. Lee. Value prediction network. *arXiv preprint arXiv:1707.03497*, 2017.

R. Ortner, O.-A. Maillard, and D. Ryabko. Selecting near-optimal approximate state representations in reinforcement learning. In *International Conference on Algorithmic Learning Theory*, pages 140–154. Springer, 2014.

I. Osband, C. Blundell, A. Pritzel, and B. Van Roy. Deep exploration via bootstrapped dqn. *arXiv preprint arXiv:1602.04621*, 2016.

H. Ossenbrink, T. Huld, A. Jäger Waldau, and N. Taylor. Photovoltaic electricity cost maps. Technical report, JRC, European Commission, 2013.

E. Parisotto, J. L. Ba, and R. Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. *arXiv preprint arXiv:1511.06342*, 2015.

R. Pascanu, Y. Li, O. Vinyals, N. Heess, L. Buesing, S. Racanière, D. Reichert, T. Weber, D. Wierstra, and P. Battaglia. Learning model-based planning from scratch. *arXiv preprint arXiv:1707.06170*, 2017.

E. Pednault, N. Abe, and B. Zadrozny. Sequential cost-sensitive decision making with reinforcement learning. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 259–268. ACM, 2002.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

X. B. Peng, G. Berseth, K. Yin, and M. van de Panne. Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning. *ACM Transactions on Graphics (Proc. SIGGRAPH 2017)*, 36 (4), 2017.

M. Petrik and B. Scherrer. Biasing approximate dynamic programming with a lower discount factor. In *Advances in neural information processing systems*, pages 1265–1272, 2009.

J. Pineau, G. Gordon, S. Thrun, et al. Point-based value iteration: An anytime algorithm for POMDPs. In *IJCAI*, volume 3, pages 1025–1032, 2003.

W. B. Powell. What you should know about approximate dynamic programming. *Naval Research Logistics (NRL)*, 56(3):239–249, 2009.

D. Precup. Eligibility traces for off-policy policy evaluation. *Computer Science Department Faculty Publication Series*, page 80, 2000.

A. Pritzel, B. Uria, S. Srinivasan, A. Puigdomènech, O. Vinyals, D. Hassabis, D. Wierstra, and C. Blundell. Neural episodic control. *arXiv preprint arXiv:1703.01988*, 2017.

M. Ranzato, S. Chopra, M. Auli, and W. Zaremba. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*, 2015.

E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, Q. Le, and A. Kurakin. Large-scale evolution of image classifiers. *arXiv preprint arXiv:1703.01041*, 2017.

S. Reed and N. De Freitas. Neural programmer-interpreters. *arXiv preprint arXiv:1511.06279*, 2015.

M. Riedmiller. Neural fitted Q iteration–first experiences with a data efficient neural reinforcement learning method. In *Machine Learning: ECML 2005*, pages 317–328. Springer, 2005.

S. Ross, J. Pineau, B. Chaib-draa, and P. Kreitmann. A bayesian approach for learning and planning in partially observable markov decision processes. *The Journal of Machine Learning Research*, 12: 1729–1770, 2011.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5 (3):1, 1988.

G. A. Rummery and M. Niranjan. *On-line Q-learning using connectionist systems*. University of Cambridge, Department of Engineering, 1994.

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

A. A. Rusu, M. Vecerik, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell. Sim-to-real robot learning from pixels with progressive nets. *arXiv preprint arXiv:1610.04286*, 2016.

D. F. Salas and W. B. Powell. Benchmarking a scalable approximate dynamic programming algorithm for stochastic control of multidimensional energy storage problems. Technical report, Working Paper, Department of Operations Research and Financial Engineering, Princeton, NJ, 2013.

T. Salimans, J. Ho, X. Chen, and I. Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

A. Santoro, D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap. A simple neural network module for relational reasoning. *arXiv preprint arXiv:1706.01427*, 2017.

T. Schaul, J. Bayer, D. Wierstra, Y. Sun, M. Felder, F. Sehnke, T. Rückstieß, and J. Schmidhuber. Pybrain. *The Journal of Machine Learning Research*, 11:743–746, 2010.

T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

S. M. Schoenung. Characteristics and technologies for long-vs. short-term energy storage. *United States Department of Energy*, 2001.

J. Schulman, S. Levine, P. Abbeel, M. I. Jordan, and P. Moritz. Trust region policy optimization. In *ICML*, pages 1889–1897, 2015a.

J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015b.

J. Schulman, J. Ho, C. Lee, and P. Abbeel. Learning from demonstrations through the use of non-rigid registration. In *Robotics Research*, pages 339–354. Springer, 2016.

J. Schulman, P. Abbeel, and X. Chen. Equivalence between policy gradients and soft q-learning. *arXiv preprint arXiv:1704.06440*, 2017.

D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. Deterministic policy gradient algorithms. In *ICML*, 2014.

D. Silver, A. Huang, C. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016a.

D. Silver, H. van Hasselt, M. Hessel, T. Schaul, A. Guez, T. Harley, G. Dulac-Arnold, D. Reichert, N. Rabinowitz, A. Barreto, et al. The predictron: End-to-end learning and planning. *arXiv preprint arXiv:1612.08810*, 2016b.

S. Singh, R. L. Lewis, A. G. Barto, and J. Sorg. Intrinsically motivated reinforcement learning: An evolutionary perspective. *IEEE Transactions on Autonomous Mental Development*, 2(2):70–82, 2010.

S. P. Singh and R. S. Sutton. Reinforcement learning with replacing eligibility traces. *Machine learning*, 22(1-3):123–158, 1996.

S. P. Singh, T. S. Jaakkola, and M. I. Jordan. Learning without state-estimation in partially observable markovian decision processes. In *ICML*, pages 284–292, 1994.

E. J. Sondik. The optimal control of partially observable markov processes over the infinite horizon: Discounted costs. *Operations research*, 26(2):282–304, 1978.

N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1): 1929–1958, 2014.

B. C. Stadie, S. Levine, and P. Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv preprint arXiv:1507.00814*, 2015.

Y. Sun, F. Gomez, and J. Schmidhuber. Planning to be surprised: Optimal bayesian exploration in dynamic environments. In *Artificial General Intelligence*, pages 41–51. Springer, 2011.

P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.

M. Šúri, T. Cebecauer, and A. Skoczek. Solargis: solar data and online applications for pv planning and performance assessment. In *26th European photovoltaics solar energy conference*, 2011.

A. Sutera, A. Joly, V. François-Lavet, Z. A. Qiu, G. Louppe, D. Ernst, and P. Geurts. Simple connectome inference from partial correlation statistics in calcium imaging. In *Neural Connectomics*, pages 23–34, 2014.

R. S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the seventh international conference on machine learning*, pages 216–224, 1990.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction (2nd Edition, in progress)*. MIT Press, 2017.

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

C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. *arXiv preprint arXiv:1602.07261*, 2016.

A. Tamar, S. Levine, P. Abbeel, Y. WU, and G. Thomas. Value iteration networks. In *Advances in Neural Information Processing Systems*, pages 2146–2154, 2016.

B. Tanner and A. White. Rl-glue: Language-independent software for reinforcement-learning experiments. *The Journal of Machine Learning Research*, 10:2133–2136, 2009.

P. S. Thomas and E. Brunskill. Data-efficient off-policy policy evaluation for reinforcement learning. In *International Conference on Machine Learning*, 2016.

S. B. Thrun. Efficient exploration in reinforcement learning. 1992.

H. Tieleman. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 2012.

J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. *arXiv preprint arXiv:1703.06907*, 2017.

J. N. Tsitsiklis and B. Van Roy. An analysis of temporal-difference learning with function approximation. *Automatic Control, IEEE Transactions on*, 42(5):674–690, 1997.

T. S. Ustun, C. Ozansoy, and A. Zayegh. Recent developments in microgrids and example cases around the world-a review. *Renewable and Sustainable Energy Reviews*, 15(8):4030–4041, 2011.

H. Van Hasselt, A. Guez, and S. David. Deep reinforcement learning with double Q-learning. *arXiv preprint arXiv:1509.06461*, 2015.

V. N. Vapnik. Statistical learning theory. adaptive and learning systems for signal processing, communications, and control, 1998.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

A. Vezhnevets, V. Mnih, S. Osindero, A. Graves, O. Vinyals, J. Agapiou, et al. Strategic attentive writer for learning macro-actions. In *Advances in Neural Information Processing Systems*, pages 3486–3494, 2016.

J. X. Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J. Z. Leibo, R. Munos, C. Blundell, D. Kumaran, and M. Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

Z. Wang, N. de Freitas, and M. Lanctot. Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*, 2015.

C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

T. Weber, S. Racanière, D. P. Reichert, L. Buesing, A. Guez, D. J. Rezende, A. P. Badia, O. Vinyals, N. Heess, Y. Li, et al. Imagination-augmented agents for deep reinforcement learning. *arXiv preprint arXiv:1707.06203*, 2017.

S. Whiteson and P. Stone. Evolutionary function approximation for reinforcement learning. *Journal of Machine Learning Research*, 7(May): 877–917, 2006.

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

A. P. Wolfe. Pomdp homomorphisms. In *NIPS RL Workshop*, 2006.

K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057, 2015.

Y. You, X. Pan, Z. Wang, and C. Lu. Virtual to real reinforcement learning for autonomous driving. *arXiv preprint arXiv:1704.03952*, 2017.

C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. *arXiv preprint arXiv:1609.05143*, 2016.

B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

LIST OF FIGURES

LIST OF TABLES

This dissertation has been submitted in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in Computer Science.

This version of the manuscript is pending the approval of the jury.