Master's Thesis : Deep Reinforcement Learning with Applications to the Renewable Energy Transition

Auteur : Bolland, Adrien
Promoteur(s) : Ernst, Damien; Wehenkel, Louis
Faculté : Faculté des Sciences appliquées
Diplôme : Master : ingénieur civil électricien, à finalité spécialisée en "signal processing and intelligent robotics"
Année académique : 2019-2020
URI/URL : http://hdl.handle.net/2268.2/8742

–University of Liège –
Faculty of Applied Science

Deep Reinforcement Learning with Applications to the Renewable Energy Transition

Master Thesis Electrical Engineering

Bolland Adrien — 20150877

Academic year 2019-2020

Acknowledgments

First of all, I would like to thank Ioannis Boukas, with whom I have worked in close collaboration over the past year. I wish him good luck for the final sprint of his PhD. thesis. I would also like to thank my promoter, Professor Damien Ernst, for his help and for introducing me to the fascinating world of research. I should also thank Mathias Berger and François Cornet for their help and for carefully reviewing this work. Many thanks to the professors of the University of Liege for providing such a stimulating learning environment. Finally, I would like to thank those persons the closest to me: my friends, my sister and, of course, my mother for teaching me the value of education, her support, and her encouragement.

Summary

The major integration of variable energy resources is expected to shift a large proportion of energy exchanges closer to real-time, where more accurate forecasts are available. In this context, short-term electricity markets, and in particular the intraday market, are considered a suitable trading floor for these exchanges to occur. A key component for the successful integration of renewable energy sources is the use of energy storage. In the first part of this work, we propose a novel modelling framework for the strategic participation of energy storage in the European continuous intraday energy market where exchanges occur through a centralized order book. The goal of the storage device operator is the maximization of the profits received over the entire trading horizon, while taking into account the operational constraints of the unit. The sequential decision-making problem of trading in the intraday market is modelled as a Markov Decision Process. An asynchronous distributed version of the fitted Q iteration algorithm is chosen for solving this problem owing to its sample efficiency. The large and variable number of existing orders in the order book motivates the use of high level actions and an alternative state representation. Historical data are used for the generation of a large number of artificial trajectories in order to address exploration issues during the learning process. The resulting policy is back-tested and compared against a benchmark strategy that is the current industrial standard. Results indicate that the agent converges to a policy that achieves, on average, higher total revenues than the benchmark strategy.

In the second part of this work, we generalise the direct policy search algorithms to an algorithm we call Direct Environment Search with (projected stochastic) Gradient Ascent (DESGA). The latter can be used to jointly learn a Reinforcement Learning (RL) environment and a policy with maximal expected return over a joint hypothesis space of environments and policies. We illustrate the performance of DESGA on two benchmarks. First, we consider a parametric space of mass spring damper environments. Then, we use our algorithm for optimizing the size of the components and the operation of a small-scale and autonomous energy system, i.e. a solar off-grid microgrid, composed of photovoltaic panels, batteries, etc. The results highlight the excellent performances of the DESGA algorithm.

Contents

1 Introduction

The arrival of renewable energies in the electricity markets has made the work of electricity traders extremely difficult. In order to reduce uncertainty of production, traders have had to engage with short-term markets such as the intraday market [Karanfil and Li, 2017]. Nevertheless, uncertainty remains high and the need to make quick decisions, based on considerable data, seems necessary. Renewable energies have also given rise to energy communities and microgrid networks. These networks consist of local electrical sub-grids shared by users. They are equipped with renewable production units, often solar, batteries for storage, and sometimes with fossil fuel generators for backup. The aim of these networks is to promote local production and self-consumption. Here too, complex decision-making problems arise. First, the different production units and storage devices have to be managed in order to supply the consumers at the lowest possible cost while still ensuring a security of supply. Second, many questions relating to the optimal sizing of these microgrids also arise, including what the optimal investments to be made are to minimize energy costs. These investment problems are closely related to the problems of optimal operation since the value of an investment is determined by the way it is operated [François-Lavet et al., 2016a, François-Lavet et al., 2016b].

The common thread of this master thesis is to address the above-mentioned decision-making problems using Artificial Intelligence (AI) and, in particular, Deep Reinforcement Learning (Deep RL) techniques. These techniques consist of using reinforcement learning algorithms with deep neural nets architecture so as to be able to tackle decision problems with high-dimensional/continuous state-action spaces [Schulman et al., 2015, Arulkumaran et al., 2017].

In the first part of the thesis, the problem of decision making in intraday markets is addressed. In this part, an automatic trader model is developed. This model allows one to represent the interactions of a trader on the market as well as the influence of the actions it takes. The agent's decisions are studied, and a trading policy is learned from historical data by using a Deep RL algorithm. This policy yields higher profits than a greedy policy which tries to maximize instantaneous profits without considering the future evolution of the market. This research has been conducted with several members of the University of Liège and of the Engie trading group. The results have been reported in a scientific publication [Boukas et al., 2020].

Within the framework of this research, it was necessary to compare the revenues generated by our trading agent with the maximum possible revenues that one could obtain on the current trading day. Applying a Monte-Carlo Tree Search (MCTS) theoretically allows us to find these maximum revenues. However, MCTS algorithms require the generation of a significant number of trajectories, and these were computationally too costly to generate to obtain a good solution within a reasonable amount of time. This was the reason for trying to identify, automatically, an MCTS algorithm performing at best for a given problem so as to minimize the number of trajectories it would need to reach an optimal solution. This search for the best MCTS algorithm

was formalised as a problem of optimizing, simultaneously, a policy and an environment, under some appropriate constraints on the set of possible environments. Surprisingly, this problem had not been addressed by the reinforcement learning community. Furthermore, the problem of sizing a microgrid and numerous other problems in engineering can be cast under this form.

This is why the second part of the master thesis focusses solely on this problem. This research led to the development of an extension of a gradient-based policy search algorithm, namely the REINFORCE algorithm [Williams, 1992], to this problem of searching for a pair of policy-environments leading to high expected returns. The new algorithm, named Direct Environment Search with (projected stochastic) Gradient Ascent, or DESGA for short, was tested on a modelled example where one needs to control a mass attached to a spring and on a more complex problem related to the dimensioning of an electrical solar microgrid. This work led to a scientific publication [Bolland et al., 2020].

The rest of this manuscript appends, first, the scientific publications about intraday trading I co-wrote [Boukas et al., 2020], and, second, the publication about the DESGA algorithm for which I am the first author [Bolland et al., 2020].

Bibliography

[Arulkumaran et al., 2017] Arulkumaran, K., Deisenroth, M. P., Brundage, M., and Bharath, A. A. (2017). A brief survey of deep reinforcement learning. *arXiv preprint arXiv:1708.05866*.

[Bolland et al., 2020] Bolland, A., Boukas, I., Cornet, F., Berger, M., and Ernst, D. (2020). Learning optimal environments using projected stochastic gradient ascent. *arXiv preprint arXiv:2006.01738*.

[Boukas et al., 2020] Boukas, I., Ernst, D., Théate, T., Bolland, A., Huynen, A., Buchwald, M., Wynants, C., and Cornélusse, B. (2020). A deep reinforcement learning framework for continuous intraday market bidding. *arXiv preprint arXiv:2004.05940*.

[François-Lavet et al., 2016a] François-Lavet, V., Gemine, Q., Ernst, D., and Fonteneau, R. (2016a). 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.

[François-Lavet et al., 2016b] François-Lavet, V., Taralla, D., Ernst, D., and Fonteneau, R. (2016b). Deep reinforcement learning solutions for energy microgrids management. In *European Workshop on Reinforcement Learning (EWRL 2016)*.

[Karanfil and Li, 2017] Karanfil, F. and Li, Y. (2017). The role of continuous intraday electricity markets: The integration of large-share wind power generation in denmark. *The Energy Journal*, 38(2).

[Schulman et al., 2015] Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015). Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897.

[Williams, 1992] Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

2 A Deep Reinforcement Learning Framework for Continuous Intraday Market Bidding

A Deep Reinforcement Learning Framework for Continuous Intraday Market Bidding

Ioannis Boukas [1] **Damien Ernst** [1] **Thibaut Théate** [1] **Adrien Bolland** [1] **Alexandre Huynen** [2] **Martin Buchwald** [2]
Christelle Wynants [2] **Bertrand Cornélusse** [1]

Abstract

The large integration of variable energy resources is expected to shift a large part of the energy exchanges closer to real-time, where more accurate forecasts are available. In this context, the short-term electricity markets and in particular the intraday market are considered a suitable trading floor for these exchanges to occur. A key component for the successful renewable energy sources integration is the usage of energy storage. In this paper, we propose a novel modelling framework for the strategic participation of energy storage in the European continuous intraday market where exchanges occur through a centralized order book. The goal of the storage device operator is the maximization of the profits received over the entire trading horizon, while taking into account the operational constraints of the unit. The sequential decision-making problem of trading in the intraday market is modelled as a Markov Decision Process. An asynchronous distributed version of the fitted Q iteration algorithm is chosen for solving this problem due to its sample efficiency. The large and variable number of the existing orders in the order book motivates the use of high-level actions and an alternative state representation. Historical data are used for the generation of a large number of artificial trajectories in order to address exploration issues during the learning process. The resulting policy is back-tested and compared against a benchmark strategy that is the current industrial standard. Results indicate that the agent converges to a policy that achieves in average higher total revenues than the benchmark strategy.

[1]Department of Electrical Engineering and Computer Science, University of Liège, Liège, Belgium [2]Market Modeling and Market View, ENGIE, Brussels, Belgium. Correspondence to: Ioannis Boukas <ioannis.boukas@uliege.be>.

1. Introduction

The vast integration of renewable energy resources (RES) into (future) power systems, as directed by the recent worldwide energy policy drive (The European Commission, 2017), has given rise to challenges related to the security, sustainability and affordability of the power system ("The Energy Trilemma"). The impact of high RES penetration on the modern short-term electricity markets has been the subject of extensive research over the last few years. Short-term electricity markets in Europe are organized as a sequence of trading opportunities where participants can trade energy in the day-ahead market and can later adjust their schedule in the intraday market until the physical delivery. Deviations from this schedule are then corrected by the transmission system operator (TSO) in real time and the responsible parties are penalized for their imbalances (Meeus & Schittekatte, 2017).

Imbalance penalties serve as an incentive for all market participants to accurately forecast their production and consumption and to trade based on these forecasts (Scharff & Amelin, 2016). Due to the variability and the lack of predictability of RES, the output planned in the day-ahead market may differ significantly from the actual RES output in real time (Karanfil & Li, 2017). Since the RES forecast error decreases substantially with a shorter prediction horizon, the intraday market allows RES operators to trade these deviations whenever an improved forecast is available (Borggrefe & Neuhoff, 2011). As a consequence, intraday trading is expected to reduce the costs related to the reservation and activation of capacity for balancing purposes. The intraday market is therefore a key aspect towards the cost-efficient RES integration and enhanced system security of supply.

Owing to the fact that commitment decisions are taken close to real time, the intraday market is a suitable market floor for the participation of flexible resources (i.e. units able to rapidly increase or decrease their generation/consumption). However, fast-ramping thermal units (e.g. gas power plants) incur a high cost when forced to modify their output, to

operate in part load, or to frequently start up and shut down. The increased cost related to the cycling of these units will be reflected to the offers in the intraday market (Pérez Arriaga & Knittel et al, 2016). Alternatively, flexible storage devices (e.g. pumped hydro storage units or batteries) with low cycling and zero fuel cost can offer their flexibility at a comparatively low price, close to the gate closure. Hence, they are expected to play a key role in the intraday market.

1.1. Intraday markets in Europe

In Europe, the intraday markets are organized in two distinct designs, namely auction-based or continuous trading.

In auction-based intraday markets, participants can submit their offers to produce or consume energy at a certain time slot until gate closure. After the gate closure, the submitted offers are used to form the aggregate demand and supply curves. The intersection of the aggregate curves defines the clearing price and quantity (Neuhoff et al., 2016). The clearing rule is uniform pricing, according to which there is only one clearing price at which all transactions occur. Participants are incentivized to bid at their marginal cost since they are paid at the uniform price. This mechanism increases price transparency, although it leads to inefficiencies, since imbalances after the gate closure can no longer be traded (Hagemann, 2015).

In continuous intraday (CID) markets, participants can submit at any point during the trading session orders to buy or to sell energy. The orders are treated according to the first come first served (FCFS) rule. A transaction occurs as soon as the price of a new "Buy" ("Sell") order is equal or higher (lower) than the price of an existing "Sell" ("Buy") order. Each transaction is settled following the pay-as-bid principle, stating that the transaction price is specified by the oldest order of the two present in the order book. Unmatched orders are stored in the order book and are accessible to all market participants. The energy delivery resolution offered by the CID market in Europe ranges between hourly, 30-minute and 15-minute products, and the gate closure takes place between five and 60 minutes before actual delivery. Continuous trading gives the opportunity to market participants to trade imbalances as soon as they appear (Hagemann, 2015). However, the FCFS rule is inherently associated with lower allocative inefficiency compared to auction rules. This implies that, depending on the time of arrival of the orders, some trades with a positive welfare contribution may not occur while others with negative welfare contribution may be realised (Henriot, 2014). It is observed that a combination of continuous and auction-based intraday markets can increase the market efficiency in terms of liquidity and market depth, and results in reduced price volatility (Neuhoff et al., 2016).

In practice, the available contracts ("Sell" and "Buy" orders) can be categorized into three types:

- The market order, where no price limit is specified (the order is matched at the best price)
- The limit order, which contains a price limit and can only be matched at that or at a better price
- The market sweep order, which is executed immediately (fully or partially) or gets cancelled.

Limit orders may appear with restrictions related to their execution and their validity. For instance, an order that carries the specification *Fill or Kill* should either be fully and immediately executed or cancelled. An order that is specified as *All or Nothing* remains in the order book until it is entirely executed (Balardy, 2017a).

The European Network Codes and specifically the capacity allocation and congestion management guidelines (Meeus & Schittekatte, 2017) (CACM GL) suggest that continuous trading should be the main intraday market mechanism. Complementary regional intraday auctions can also be put in place if they are approved by the regulatory authorities (Meeus & Schittekatte, 2017). To that direction, the Cross-Border Intraday (XBID) Initiative (Spot, 2018) has enabled continuous cross-border intraday trading across Europe. Participants of each country have access to orders placed from participants of any other country in the consortium through a centralized order book, provided that there is available cross-border capacity.

1.2. Bidding strategies in literature

The strategic participation of power producers in short-term electricity markets has been extensively studied in the literature. In order to co-optimise the decisions made in the sequential trading floors from day-ahead to real time the problem has been traditionally addressed using multi-stage stochastic optimisation. Each decision stage corresponds to a trading floor (i.e. day-ahead, capacity markets, real-time), where the final decisions take into account uncertainty using stochastic processes. In particular, the influence that the producer may have on the market price formation leads to the distinction between "price-maker" and "price-taker" and results in a different modelling of the uncertainty.

In (Baillo et al., 2004), the optimisation of a portfolio of generating assets over three trading floors (i.e. the day-ahead, the adjustment and the reserves market) is proposed, where the producer is assumed to be a "price-maker". The offering strategy of the producer is a result of the stochastic residual demand curve as well as the behaviour of the rest of the market players. On the contrary, a "price-taker" producer is considered in (Plazas et al., 2005) for the first two stages of the problem studied, namely the day-ahead and the automatic generation control (AGC) market. However, since the third-stage (balancing market) traded volumes are small,

the producer can negatively affect the prices with its participation. Price scenarios are generated using ARIMA models for the two first stages, whereas for the third stage a linear curve with negative slope is used to represent the influence of the producer's offered capacity on the market price.

Hydro-power plant participation in short-term markets accounting for the technical constraints and several reservoir levels is formulated and solved in (Fleten & Kristoffersen, 2007). Optimal bidding curves for the participation of a "price-taker" hydro-power producer in the Nordic spot market are derived accounting for price uncertainty. In (Boomsma et al., 2014), the bidding strategy of a two-level reservoir plant is casted as a multi-stage stochastic program in order to represent the different sequential trading floors, namely the day-ahead spot market and the hour-ahead balancing market. The effects of coordinated bidding and the "price-maker" versus "price-taker" assumptions on the generated profits are evaluated. In (PandŽić et al., 2013), bidding strategies for a virtual power plant (VPP) buying and selling energy in the day-ahead and the balancing market in the form of a multi-stage stochastic optimisation are investigated. The VPP aggregates a pumped hydro energy storage (PHES) unit as well as a conventional generator with stochastic intermittent power production and consumption. The goal of the VPP operator is the maximization of the expected profits under price uncertainty.

In these approaches, the intraday market is considered as auction-based and it is modelled as a single recourse action. For each trading period, the optimal offered quantity is derived according to the realization of various stochastic variables. However, in reality, for most European countries, according to the EU Network Codes (Meeus & Schittekatte, 2017), modern intraday market trading will primarily be a continuous process.

The strategic participation in the CID market is investigated for the case of an RES producer in (Henriot, 2014) and (Garnier & Madlener, 2015). In both works, the problem is formulated as a sequential decision-making process, where the operator adjusts its offers during the trading horizon, according to the RES forecast updates for the physical delivery of power. Additionally, in (Gönsch & Hassler, 2016) the use of a PHES unit is proposed to undertake energy arbitrage and to offset potential deviations. The trading process is formulated as a Markov Decision Process (MDP) where the future commitment decision in the market is based on the stochastic realization of the intraday price, the imbalance penalty, the RES production and the storage availability.

The volatility of the CID prices, along with the quality of the forecast updates, are found to be key factors that influence the degree of activity and success of the deployed bidding strategies (Henriot, 2014). Therefore, the CID prices and the forecast errors are considered as correlated stochastic processes in (Garnier & Madlener, 2015). Alternatively, in (Henriot, 2014), the CID price is constructed as a linear function of the offered quantity with an increasing slope as the gate closure approaches. In this way, the scarcity of conventional units approaching real time is reflected. In (Gönsch & Hassler, 2016), real weather data and market data are used to simulate the forecast error and CID price processes.

For the sequential decision-making problem in the CID market, the offered quantity of energy is the decision variable to be optimised (Garnier & Madlener, 2015). The optimisation is carried out using Approximate Dynamic Programming (ADP) methods, where a parameterised policy is obtained based on the observed stochastic processes for the price, the RES error and the level of the reservoir (Gönsch & Hassler, 2016). The ADP approach presented in (Gönsch & Hassler, 2016) is compared in (Hassler, 2017) to some threshold-based heuristic decision rules. The parameters are updated according to simulation-based experience and the obtained performance is comparable to the ADP algorithm. The obtained decision rules are intuitively interpretable and are derived efficiently through simulation-based optimisation.

The bidding strategy deployed by a storage device operator participating in a slightly different real-time market organized by NYISO is presented in (Jiang & Powell, 2014). In this market, the commitment decision is taken one hour ahead of real-time and the settlements occur intra-hour every five minutes. In this setting, the storage operator selects two price thresholds at which the intra-hour settlements occur. The problem is formulated as an MDP and is solved using an ADP algorithm that exploits a particular monotonicity property. A distribution-free variant that assumes no knowledge of the price distribution is proposed. The optimal policy is trained using historical real-time price data.

Even though the focus of the mentioned articles lies on the CID market, the trading decisions are considered to take place in discrete time-steps. A different approach is presented in (Aïd et al., 2016), where the CID market participation is modelled as a continuous time process using stochastic differential equations (SDE). The Hamilton Jacobi Bellman (HJB) equation is used for the determination of the optimal trading strategy. The goal is the minimization of the imbalance cost faced by a power producer arising from the residual error between the RES production and demand. The optimal trading rate is derived assuming a stochastic process for the market price using real market data and the residual error.

In the approaches presented so far, the CID price is modelled as a stochastic process assuming that the participating agent is a "price-taker". However, in the CID market, this assumption implies that the CID market is liquid and the price at which one can buy or sell energy at a given time

are similar or the same. This assumption does not always hold, since the mean bid-ask spread in a trading session in the German intraday market for 2015 was several hundred times larger than the tick-size (i.e. the minimum price movement of a trading instrument) (Balardy, 2017b). It is also reported in the same study that the spread decreases as trading approaches the gate closure.

An approach that explicitly considers the order book is presented in (Bertrand & Papavasiliou, 2019). A threshold-based policy is used to optimise the bid acceptance for storage units participating in the CID market. A collection of different factors such as the time of the day are used for the adaptation of the price thresholds. The threshold policy is trained using a policy gradient method (REINFORCE) and the results show improved performance against the "rolling intrinsic" benchmark. In this paper, we present a methodology that serves as a wrapper around the existing industrial state of the art. We provide a generic modelling framework for the problem and we elaborate on all the assumptions that allow the formulation of the problem as an MDP. We solve the resulting problem using a value function approximation method.

1.3. Contribution of the paper

In this paper, we focus on the sequential decision-making problem of a storage device operator participating in the CID market. Firstly, we present a novel modelling framework for the CID market, where the trading agents exchange energy via a centralized order book. Each trading agent is assumed to dynamically select the orders that maximize its benefits throughout the trading horizon. In contrast to the existing literature, all the available orders are considered explicitly with the intention to uncover more information at each trading decision. The liquidity of the market and the influence of each agent on the price are directly reflected in the observed order book, and the developed strategies can adapt accordingly. Secondly, we model explicitly the dynamics of the storage system. Finally, the resulting problem is cast as an MDP.

The intraday trading problem of a storage device is solved using Deep Reinforcement Learning techniques, specifically an asynchronous distributed variant of the fitted Q iteration RL algorithm with deep neural networks as function approximators (Ernst et al., 2005). Due to the high-dimensionality and the dynamically evolving size of the order book, we motivate the use of high-level actions and a constant size, low-dimensional state representation. The agent can select between trading and idling. The goal of the selected actions is the identification of the opportunity cost of trading given the state of the order book observed and the maximization of the total return over the trading horizon. The resulting optimal policy is evaluated using real data from the German ID market (EPEXSPOT, 2017). In summary, the contributions of this work are the following:

- We model the CID market trading process as an MDP where the energy exchanges occur explicitly through a centralized order book. The trading policy is adapted to the state of the order book.
- The operational constraints of the storage device are considered explicitly.
- A state space reduction is proposed in order to deal with the size of the order book.
- A novel representation of high-level actions is used to identify the opportunity cost between trading and idling.
- The fitted Q iteration algorithm is used to find a time-variant policy that maximizes the total cumulative profits collected.
- Artificial trajectories are produced using historical data from the German CID market in order to address exploration issues.

1.4. Outline of the paper

The rest of the paper is organized as follows. In Section 2, the CID market trading framework is presented. The interaction of the trading agents via a centralized order book is formulated as a dynamic process. All the available information for an asset trading agent is detailed and the objective is defined as the cumulative profits. In Section 3, all the assumptions necessary to formulate the bidding process in the CID market as an MDP are listed. The methodology utilised to find an optimal policy that maximizes the cumulative profits of the proposed MDP is detailed in Section 4. A case study using real data from the German CID market is performed in Section 5. The results as well as considerations about limitations of the developed methodology are discussed in Section 6. Finally, conclusions of this work are drawn and future recommendations are provided in Section 7. A detailed nomenclature is provided at the Appendix A.

2. Continuous Intraday Bidding process

2.1. Continuous Intraday market design

The participation in the CID market is a continuous process similar to the stock exchange. Each market product $x \in X$, where X is the set of all available products, is defined as the physical delivery of energy in a pre-defined time slot. The time slot corresponding to product x is defined by its starting point $t_{delivery}(x)$ and its duration $\lambda(x)$. The trading process for time slot x opens at $t_{open}(x)$ and closes at $t_{close}(x)$.

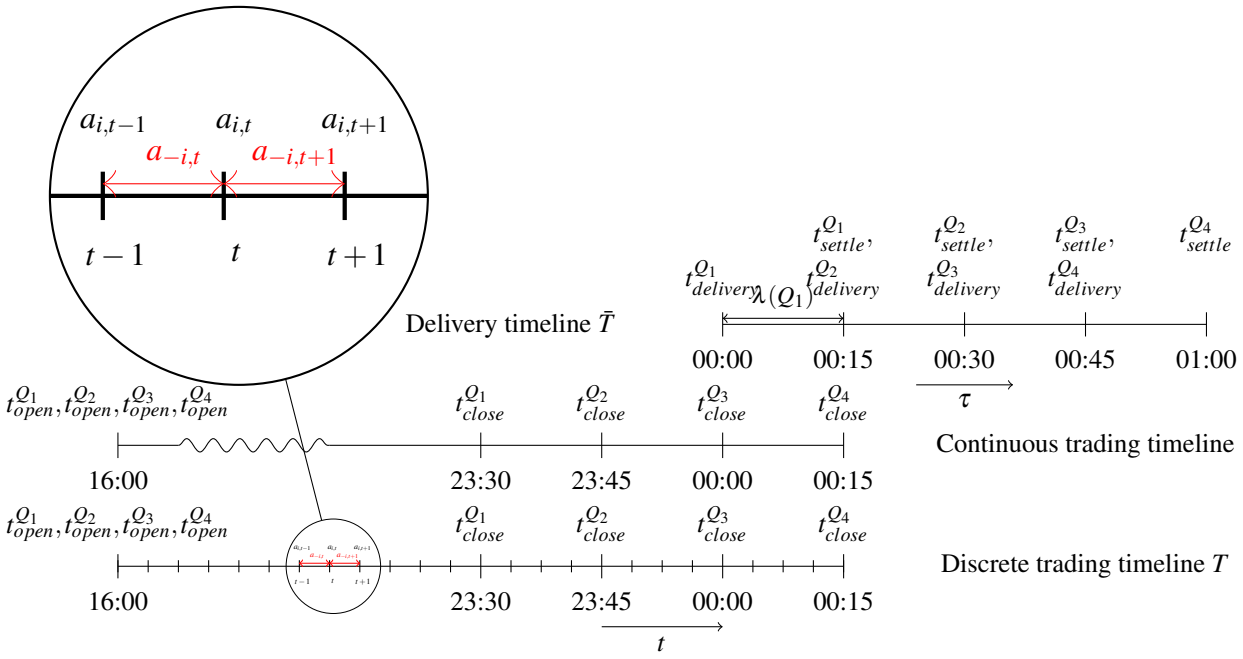


Figure 1. Trading (continuous and discrete) and delivery timelines for products Q_1 to Q_4

During the time interval $t \in [t_{open}(x), t_{close}(x)]$, a participant can exchange energy with other participants for the lagged physical delivery during the interval $\delta(x)$, with:

$$\delta(x) = \left[t_{delivery}(x), t_{delivery}(x) + \lambda(x)\right].$$

The exchange of energy takes place through a centralized order book that contains all the unmatched orders o_j, where $j \in N_t$ corresponds to a unique index that every order receives upon arrival. The set $N_t \subseteq \mathbb{N}$ gathers all the unique indices of the orders available at time t. We denote the status of the order book at time t by $O_t = (o_j, \forall j \in N_t)$. As time progresses new orders appear and existing ones are either accepted or cancelled.

Trading for a set of products is considered to start at the gate opening of the first product and to finish at the gate closure of the last product. More formally, considering an ordered set of available products $X = \{Q_1, ...Q_{96}\}$, the corresponding trading horizon is defined as $T = [t_{open}(Q_1), t_{close}(Q_{96})]$. For instance, in the German CID market, trading of hourly (quarterly) products for day D opens at 3 pm (4 pm) of day $D-1$ respectively. For each product x, the gate closes 30 minutes before the actual energy delivery at $t_{delivery}(x)$. The timeline for trading products Q_1 to Q_4 that correspond to the physical delivery in 15-minute time slots from 00:00 until 01:00, is presented in Figure 1. It can be observed that the agent can trade for all products until 23:30. After each subsequent gate closure the number of available products decreases and the commitment for the corresponding time slot is defined. Potential deviations during the physical delivery of energy are penalized in the imbalance market.

2.2. Continuous Intraday market environment

As its name indicates, the CID market is a continuous environment. In order to solve the trading problem presented in this paper, it has been decided to perform a relevant discretisation operation. As shown in Figure 1, the trading timeline is discretised in a high number of time-steps of constant duration Δt. Each discretised trading interval for product x can be denoted by the set of time-steps $T(x) = \left\{t_{open}(x), t_{open}(x) + \Delta t, ..., t_{close}(x) - \Delta t, t_{close}(x)\right\}$. Then, the discrete-time trading opportunities for the entire set of products X can be modelled such that the time-steps are defined as $t \in T = \bigcup_{x \in X} T(x)$. In the following, for the sake of clarity, the increment (decrement) operation $t+1$ ($t-1$) will be used to model the discrete transition from time-step t to time-step $t + \Delta t$ ($t - \Delta t$).

It is important to note that in theory the discretisation operation leads to suboptimalities in the decision-making process. However, as the discretisation becomes finer ($\Delta t \to 0$), the decisions taken can be considered near-optimal. Increasing the granularity of the decision time-line results in an increase of the number of decisions that can be taken and hence, the size of the decision-making problem. Thus, there is a clear trade-off between complexity and quality of the resulting decisions when using a finite discretisation.

Let X_t denote the set of available products at time-step $t \in T$

such that:

$$X_t = \{x | x \in X, t \leq t_{close}(x)\}.$$

We define the state of the CID market environment at time-step t as $s_t^{OB} = O_t \in S^{OB}$. The state contains the observation of the order book at time-step $t \in T$ i.e. the unmatched orders for all the available products $x \in X_t \subset X$.

A set of n agents $I = \{1, 2, ..., n\}$ are continuously interacting in the CID environment exchanging energy. Each agent $i \in I$ can express its willingness to buy or sell energy by posting at instant t a set of new orders $a_{i,t} \in A_i$ in the order book, which results in the joint action $a_t = (a_{1,t}, ..., a_{n,t}) \in \prod_{i=1}^{n} A_i$.

The process of designing the set of new orders $a_{i,t}$ for agent i at instant t consists, for each new order, in determining the product $x \in X_t$, the side of the order $y \in \{\text{“}Sell\text{”}, \text{“}Buy\text{”}\}$, the volume $v \in \mathbb{R}^+$, the price level $p \in [p_{min}, p_{max}]$ of each unit offered to be produced or consumed, and the various validity and execution specifications $e \in E$. The index of each new order j belongs to the set $j \in N_t'$.

The set of new orders is defined as $a_{i,t} = ((x_j, y_j, v_j, p_j, e_j), \forall j \in N_t' \subseteq \mathbb{N})$. We will use the notation for the joint action $a_t = (a_{i,t}, a_{-i,t})$ to refer to the action that agent i selects $a_{i,t}$ and the joint action that all other agents use $a_{-i,t} = (a_{1,t}, ..., a_{i-1,t}, a_{i+1.t}, ..., a_{n,t})$.

Table 1. Order Book for Q_1 and time slot 00:00-00:15

i	**Side**	v **[MW]**	p **[€/MWh]**	
4	“Sell”	6.25	36.3	
2	“Sell”	2.35	34.5	⟵ ask
1	“Buy”	3.15	33.8	⟵ bid
3	“Buy”	1.125	29.3	
5	“Buy”	2.5	15.9	

The orders are treated according to the first come first served (FCFS) rule. Table 1 presents an observation of the order book for product Q_1. The difference between the most expensive “Buy” order (“bid”) and the cheapest “Sell” order (“ask”) defines the bid-ask spread of the product. A deal between two counter-parties is struck when the price p_{buy} of a “Buy” order and the price p_{sell} of a “Sell” order satisfy the condition $p_{buy} \geq p_{sell}$. This condition is tested at the arrival of each new order. The volume of the transaction is defined as the minimum quantity between the “Buy” and “Sell” order ($\min(v_{buy}, v_{sell})$). The residual volume remains available in the market at the same price. As mentioned in the previous section, each transaction is settled following the pay-as-bid principle, at the price indicated by the oldest order.

Finally, at each time-step t, every agent i observes the state of the order book s_t^{OB}, performs certain actions (posting a set of new orders) $a_{i,t}$, inducing a transition which can be represented by the following equation:

$$s_{t+1}^{OB} = f(s_t^{OB}, a_{i,t}, a_{-i,t}). \tag{1}$$

2.3. Asset trading

An asset optimizing agent participating in the CID market can adjust its position for product x until the corresponding gate closure $t_{close}(x)$. However, the physical delivery of power is decided at $t_{delivery}(x)$. An additional amount of information (potentially valuable for certain players) is received during the period $\{t_{close}(x), .., t_{delivery}(x)\}$, from the gate closure until the delivery of power. Based on this updated information, an asset-trading agent may need to or have an incentive to deviate from the net contracted power in the market.

Let $v_{i,t}^{con} = (v_{i,t}^{con}(x), \forall x \in X_t) \in \mathbb{R}^{|X_t|}$, gather the volumes of power contracted by agent i for the available products $x \in X_t$ at each time-step $t \in T$. In the following, we will adopt the convention for $v_{i,t}^{con}(x)$ to be positive when agent i contracts the net volume to sell (produce) and negative when the agent contracts the volume to buy (consume) energy for product x at time-step t.

Following each market transition as indicated by equation (1), the volumes contracted $v_{i,t}^{con}$ are determined based on the transactions that have occurred. The contracted volumes $v_{i,t}^{con}$ are derived according to the FCFS rule that is detailed in (EPEXSPOT, 2019). The mathematical formulation of the clearing algorithm is provided in (Le et al., 2019). The objective function of the clearing algorithm is comprised of two terms, namely the social welfare and a penalty term modelling the price-time priority rule. The orders that maximize this objective are matched, provided that they satisfy the balancing equations and constraints related to their specifications. The clearing rule is implicitly given by:

$$v_{i,t}^{con} = clear(i, s_t^{OB}, a_{i,t}, a_{-i,t}). \tag{2}$$

We denote as $P_{i,t}^{mar}(x) \in \mathbb{R}$ the net contracted power in the market by agent i for each product $x \in X$, which is updated at every time-step $t \in T$ according to:

$$P_{i,t+1}^{mar}(x) = P_{i,t}^{mar}(x) + v_{i,t}^{con}(x). \quad \forall x \in X_t \tag{3}$$

The discretisation of the delivery timeline $\bar{T}$ is done with time-steps of duration $\Delta\tau$, equal to the minimum duration of delivery for the products considered. The discrete delivery timeline $\bar{T}$ is considered to start at the beginning of delivery of the first product τ_{init} and to finish at the end of the delivery of the last product τ_{term}. For the simple case where only four quarterly products are considered, as shown in Figure 1, the

delivery time-step is $\Delta\tau = 15min$ and the delivery timeline $\bar{T} = \{00:00, 00:15, ..., 01:00\}$, where $\tau_{init} = 00:00$ and $\tau_{term} = 01:00$. In general, when only one type of product is considered (e.g. quarterly), there is a straightforward relation between time of delivery τ and product x, since $\tau = t_{delivery}(x)$ and $\Delta\tau = \lambda(x)$. Thus, terms x or τ can be used interchangeably. For the sake of keeping the notation relatively simple, we will only consider quarterly products in the rest of the paper. In such a context, the terms $P^{mar}_{i,t}(\tau)$ or $P^{mar}_{i,t}(x)$ can be used interchangeably to denote the net contracted power in the market by agent i at trading step t for delivery time-step τ (product x).

As the trading process evolves the set of delivery time-steps τ for which the asset-optimizing can make decisions decreases as trading time t crosses the delivery time τ. Let $\bar{T}(t) \subseteq \bar{T}$ be a function that yields the subset of delivery time-steps $\tau \in \bar{T}$ that follow time-step $t \in T$ such that:

$$\bar{T}(t) = \{\tau | \tau \in \bar{T} \setminus \{\tau_{term}\}, t \leq \tau\}.$$

The participation of an asset-optimizing agent in the CID market is composed of two coupled decision processes with different timescales. First, the trading process where a decision is taken at each time-step t about the energy contracted until the gate closure $t_{close}(x)$. During this process, the agent can decide about its position in the market and create scenarios/make projections about the actual delivery plan based on its position. Second, the physical delivery decision that is taken at the time of the delivery τ or $t_{delivery}(x)$ based on the total net contracted power in the market during the trading process.

An agent i participating in the CID market is assumed to monitor the state of the order book s^{OB}_t and its net contracted power in the market $P^{mar}_{i,t}(x)$ for each product $x \in X$, which becomes fixed once the gate closure occurs at $t_{close}(x)$. Depending on the role it presumes in the market, an asset-optimizing agent is assumed to monitor all the available information about its assets. We distinguish the three following cases among the many different roles that can be played by an agent in the CID market:

- *The agent controls a physical asset that can generate and/or consume electricity*. We define as $G_{i,t}(\tau) \in [\underline{G_i}, \overline{G_i}]$ the power production level for agent i at delivery time-step τ as computed at trading step t. In a similar way, we define the power consumption level $C_{i,t}(\tau) \in [\underline{C_i}, \overline{C_i}]$, where $\underline{C_i}, \overline{C_i}, \underline{G_i}, \overline{G_i} \in \mathbb{R}^+$. We further assume that the actual production $g_{i,t}(t')$ and consumption level $c_{i,t}(t')$ during the time-period of delivery $t' \in [\tau, \tau + \Delta\tau)$, is constant for each product x such that:

$$g_{i,t}(t') = G_{i,t}(\tau), \quad (4)$$
$$c_{i,t}(t') = C_{i,t}(\tau), \quad (5)$$
$$\forall t' \in [\tau, \tau + \Delta\tau).$$

At each time-step t during the trading process, agent i can decide to adjust its generation level by $\Delta G_{i,t}(\tau)$ or its consumption level by $\Delta C_{i,t}(\tau)$. According to these adjustments the generation and consumption levels can be updated at each time-step t according to:

$$G_{i,t+1}(\tau) = G_{i,t}(\tau) + \Delta G_{i,t}(\tau), \quad (6)$$
$$C_{i,t+1}(\tau) = C_{i,t}(\tau) + \Delta C_{i,t}(\tau), \quad (7)$$
$$\forall \tau \in \bar{T}(t).$$

Let $w^{exog}_{i,t}$ denote any other relevant exogenous information to agent i such as the RES forecast, a forecast of the actions of other agents, or the imbalance prices. The computation of $\Delta G_{i,t}(\cdot)$ and $\Delta C_{i,t}(\cdot)$ depends on the market position, the technical limits of the assets, the state of the order book and the exogenous information $w^{exog}_{i,t}$. We define the residual production $P^{res}_{i,t}(\tau) \in \mathbb{R}$ at delivery time-step τ as the difference between the production and the consumption levels and can be computed by:

$$P^{res}_{i,t}(\tau) = G_{i,t}(\tau) - C_{i,t}(\tau). \quad (8)$$

We note that the amount of residual production $P^{res}_{i,t}(\tau)$ aggregates the combined effects that $G_{i,t}(\tau)$ and $C_{i,t}(\tau)$ have on the revenues made by agent i through interacting with the markets (intraday/imbalance).

The level of generation and consumption for a market period τ can be adjusted at any time-step t before the physical delivery τ, but it becomes binding when $t = \tau$. We denote as $\Delta_{i,t}(\tau)$ the deviation from the market position for each time-step τ, as scheduled at time t, after having computed the variables $G_{i,t}(\tau)$ and $C_{i,t}(\tau)$, as follows:

$$P^{mar}_{i,t}(\tau) + \Delta_{i,t}(\tau) = P^{res}_{i,t}(\tau), \quad (9)$$
$$\forall \tau \in \bar{T}(t).$$

The term $\Delta_{i,t}(\tau)$ represents the imbalance for market period τ as estimated at time t. This imbalance may evolve up to time $t = \tau$. We denote by $\Delta_i(\tau) = \Delta_{i,t=\tau}(\tau)$ the final imbalance for market period τ.

The power balance of equation (9) written for time-step $t+1$ is given by:

$$P^{mar}_{i,t+1}(\tau) + \Delta_{i,t+1}(\tau) = G_{i,t+1}(\tau) - C_{i,t+1}(\tau) \quad (10)$$
$$\forall \tau \in \bar{T}(t+1).$$

It can be observed that by substituting equations (3), (6) and (7) in equation (10) we have:

$$P^{mar}_{i,t}(\tau)+v^{con}_{i,t}(\tau)+\Delta_{i,t+1}(\tau)= G_{i,t}(\tau)+\Delta G_{i,t}(\tau)-(C_{i,t}(\tau)+\Delta C_{i,t}(\tau)) \quad \forall\tau\in\bar{T}(t). \tag{11}$$

The combination of equations (8) and (9) with equation (11) yields the update of the imbalance vector according to:

$$\Delta_{i,t+1}(\tau)=\Delta_{i,t}(\tau)+\Delta G_{i,t}(\tau)-\Delta C_{i,t}(\tau)-v^{con}_{i,t}(\tau) \quad \forall\tau\in\bar{T}(t). \tag{12}$$

- *The agent does not own any physical asset (market maker).* It is equivalent to the first case with $\underline{C_i}=\overline{C_i}=\underline{G_i}=\overline{G_i}=0$. The net imbalance $\Delta_{i,t}(\tau)$ is updated at every time-step $t\in T$ according to:

$$P^{mar}_{i,t}(\tau)+\Delta_{i,t}(\tau)=0, \quad \forall\tau\in\bar{T}(t). \tag{13}$$

- *The agent controls a storage device that can produce, store and consume energy*. We can consider an agent controlling a storage device as an agent that controls generation and production assets with specific constraints on the generation and the consumption level related to the nature of the storage device. Following this argument, let $G_{i,t}(\tau)$ ($C_{i,t}(\tau)$) refer to the level of discharging (charging) of the storage device for delivery time-step τ, updated at time t. Obviously, if $G_{i,t}(\tau)>0$ ($C_{i,t}(\tau)>0$), then we automatically have $C_{i,t}(\tau)=0$ ($G_{i,t}(\tau)=0$) since a battery cannot charge and discharge energy at the same time. In this case, agent i can decide to adjust its discharging (charging) level by $\Delta G_{i,t}(\tau)$ ($\Delta C_{i,t}(\tau)$). Let $SoC_{i,t}(\tau)$ denote the state of charge of the storage unit at delivery time-step $\tau\in\bar{T}$ as it is computed at time-step t, where $SoC_{i,t}(\tau)\in\left[\underline{SoC_i},\overline{SoC_i}\right]$. The evolution of the state of charge during the delivery timeline can be updated at decision time-step t as:

$$SoC_{i,t}(\tau+\Delta\tau)=SoC_{i,t}(\tau)+\Delta\tau\cdot\left(\eta C_{i,t}(\tau)-\frac{G_{i,t}(\tau)}{\eta}\right), \quad \forall\tau\in\bar{T}(t). \tag{14}$$

Parameter η represents the charging and discharging efficiencies of the storage unit which, for simplicity, we assume are equal. We note that for batteries, charging and discharging efficiencies may be different and depend on the charging/discharging speeds. As can be observed from equation (14), time-coupling constraints are imposed on $C_{i,t}(\tau)$ and $G_{i,t}(\tau)$ in order to ensure that the amount of energy that can be discharged during some period already exists in the storage device. Additionally, constraints associated with the maximum charging power $\overline{C_i}$ and discharging power $\overline{G_i}$, as well as the maximum and minimum energy level ($\overline{SoC_i}$, $\underline{SoC_i}$) are considered in order to model the operation of the storage device.

Equation (14) can be written for time-step $t+1$ as:

$$SoC_{i,t+1}(\tau+\Delta\tau)=SoC_{i,t+1}(\tau)+\Delta\tau\cdot\left(\eta C_{i,t+1}(\tau)-\frac{G_{i,t+1}(\tau)}{\eta}\right), \quad \forall\tau\in\bar{T}(t+1). \tag{15}$$

Combining equations (14) and (15) we can derive the updated vector of the state of charge at time-step $t+1$ depending on the decided adjustments $(\Delta G_{i,t}(\tau),\Delta C_{i,t}(\tau))$ as:

$$SoC_{i,t+1}(\tau+\Delta\tau)-SoC_{i,t+1}(\tau)= SoC_{i,t}(\tau+\Delta\tau)-SoC_{i,t}(\tau)+\Delta\tau\cdot(\eta\Delta C_{i,t}(\tau)-\frac{\Delta G_{i,t}(\tau)}{\eta}), \quad \forall\tau\in\bar{T}(t). \tag{16}$$

The state of charge $SoC_{i,t}(\tau)$ at delivery time-step τ can be updated until $t=\tau$. Let us also observe that there is a bijection between $P^{res}_{i,t}(\tau)$ and the terms $C_{i,t}(\tau)$ and $G_{i,t}(\tau)$ or, in other words, determining $P^{res}_{i,t}$ is equivalent to determining $C_{i,t}(\tau)$ and $G_{i,t}(\tau)$ and vice versa. The deviation from the committed schedule $\Delta_{i,t+1}(\tau)$ at delivery time-step τ at each time-step $t+1$ can be computed by equation (12).

All the new information arriving at time-step t for an asset-optimizing agent i (controlling a storage device) is gathered in variable:

$$s_{i,t}=(s^{OB}_t,(P^{mar}_{i,t}(\tau),\Delta_{i,t}(\tau),G_{i,t}(\tau),C_{i,t}(\tau),SoC_{i,t}(\tau),\forall\tau\in\bar{T}),w^{exog}_{i,t})\in S_i.$$

The control action applied by an asset-optimizing agent i trading in the CID market at time-step t consists of posting new orders in the CID market and adjusting its production/consumption level or equivalently its charging/discharging level for the case of the storage device.

The control actions can be summarised in variable $u_{i,t} = (a_{i,t}, (\Delta C_{i,t}(\tau), \Delta G_{i,t}(\tau), \forall \tau \in \bar{T}))$.

In this paper, we consider that the trading agent adopts a simple strategy for determining, at each time-step t, the variables $\Delta C_{i,t}(\tau), \Delta G_{i,t}(\tau)$ once the trading actions $a_{i,t}$ have been selected. In this case, the decision regarding the trading actions $a_{i,t}$ fully defines action $u_{i,t}$ and thus the notation $u_{i,t}$ will not be further used. This strategy will be referred to in the rest of the paper as the "default" strategy for managing the storage device. According to this strategy, the agent aims at minimizing any imbalances ($\Delta_{i,t+1}(\tau)$) and therefore we use the following decision rule:

$$(\Delta C_{i,t}(\tau), \Delta G_{i,t}(\tau), \forall \tau \in \bar{T}) = \arg\min \sum_{\tau \in \bar{T}} \mid \Delta_{i,t+1}(\tau) \mid, \quad \text{s.t. (2), (3), (8), (9), (12), (14).} \tag{17}$$

One can easily see that from equation (11) this decision rule is equivalent to imposing $P^{res}_{i,t+1}(\tau)$ as close as possible to $P^{mar}_{i,t+1}(\tau)$, given the operational constraints of the device. We will elaborate later in this paper on the fact that adopting such a strategy is not suboptimal in a context where the agent needs to be balanced for every market period while being an aggressor in the CID market.

For the sake of simplicity, we assume that the decision process of an asset-optimizing agent terminates at the gate closure $t_{close}(x)$ along with the trading process. Thus, the final residual production $P^{res}_i(\tau)$ for delivery time-step τ is given by $P^{res}_i(\tau) = P^{res}_{i,t=t_{close}(x)}(\tau)$. Similarly, the final imbalance is provided by $\Delta_i(\tau) = \Delta_{i,t=t_{close}(x)}(\tau)$.

Although this approach can be used for the optimisation of a portfolio of assets, in this paper, the focus lies on the case where the agent is operating a storage device. We note that this case is particularly interesting in the context of energy transition, where storage devices are expected to play a key role in the energy market.

2.4. Trading rewards

The instantaneous reward signal collected after each transition for agent i is given by:

$$r_{i,t} = R_i(t, s_{i,t}, a_{i,t}, a_{-i,t}), \tag{18}$$

where $R_i : T \times S_i \times A_1 \times ... \times A_n \to \mathbb{R}$.

The reward function R_i is composed of the following terms:

i. The trading revenues obtained from the matching process of orders at time-step t, given by ρ where ρ is a stationary function $\rho : S^{OB} \times A_1 \times ... \times A_n \to \mathbb{R}$,

ii. The imbalance penalty for deviation $\Delta_i(\tau)$ from the market position for delivery time-step τ at the imbalance price $I(\tau)$. The imbalance settlement process for product $x \in X$ (delivery time-step τ) takes place at the end of the physical delivery $t_{settle}(x)$ (i.e. at $\tau + \Delta\tau$), as presented in Figure 1. We define the imbalance settlement timeline T^{Imb}, as $T^{Imb} = \{\tau + \Delta\tau, \forall \tau \in \bar{T}\}$. The imbalance penalty[1] is only applied when time instance t is an element of the imbalance settlement timeline.

The function R_i is defined as:

$$R_i(t, s_{i,t}, a_{i,t}, a_{-i,t}) = \rho\left(s^{OB}_t, a_{i,t}, a_{-i,t}\right) + \begin{cases} \Delta_i(\tau) \cdot I(\tau) & \text{, if } t \in T^{Imb}, \\ 0 & \text{, otherwise} \end{cases}. \tag{19}$$

2.5. Trading policy

All the relevant information that summarises the past and that can be used to optimise the market participation is assumed to be contained in the history vector $h_{i,t} = (s_{i,0}, a_{i,0}, r_{i,0}, ..., s_{i,t-1}, a_{i,t-1}, r_{i,t-1}, s_{i,t}) \in H_i$. Trading agent i is assumed to select its actions following a non-anticipative history-dependent policy $\pi_i(h_{i,t}) \in \Pi$ from the set of all admissible policies Π, according to: $a_{i,t} \sim \pi_i(\cdot|h_{i,t})$.

2.6. Trading objective

The return collected by agent i in a single trajectory $\zeta = (s_{i,0}, a_{i,0}, ..., a_{i,K-1}, s_{i,K})$ of $K-1$ time-steps, given an initial state $s_{i,0} = s_i \in S_i$, which is the sum of cumulated rewards over this trajectory is given by:

$$G^{\zeta}(s_i) = \sum_{t=0}^{K-1} R_i(t, s_{i,t}, a_{i,t}, a_{-i,t}) \,|s_{i,0} = s_i. \tag{20}$$

The sum of returns collected by agent i, where each agent i is following a randomized policy $\pi_i \in \Pi$ are consequently given by:

$$V^{\pi_i}(s_i) = \mathop{\mathbb{E}}_{a_{i,t} \sim \pi_i, a_{-i,t} \sim \pi_{-i}} \left\{ \sum_{t=0}^{K-1} R_i(t, s_{i,t}, a_{i,t}, a_{-i,t}) \,|s_{i,0} = s_i \right\}. \tag{21}$$

The goal of the trading agent i is to identify an optimal policy $\pi^*_i \in \Pi$ that maximizes the expected sum of rewards collected along a trajectory. An optimal policy is obtained by:

$$\pi^*_i = \arg\max_{\pi_i \in \Pi} V^{\pi_i}(s_i). \tag{22}$$

[1] The imbalance price $I(\tau)$ is defined by a process that depends on a plethora of factors among which is the net system imbalance during delivery period τ, defined by the imbalance volumes of all the market players ($\sum^I \Delta_i(\tau)$). For the sake of simplicity we will assume that it is randomly sampled from a known distribution over prices that is not conditioned on any variable.

3. Reinforcement Learning Formulation

In this section, we propose a series of assumptions that allow us to formulate the previously introduced problem of a storage device operator trading in the CID market using a reinforcement learning (RL) framework. Based on these assumptions, the decision-making problem is cast as an MDP; the action space is tailored in order to represent a particular market player and additional restrictions on the operation of the storage device are introduced.

3.1. Assumptions on the decision process

Assumption 1 (*Behaviour of the other agents*). The other agents $-i$ interact with the order book in between two discrete time-steps in such a way that agent i can be considered independent when interacting with the CID market at each time-step t. Moreover, it is assumed that these actions $a_{-i,t}$ depend strictly on the history of order book states s_{t-1}^{OB} and thus by extension on the history $h_{i,t-1}$ for every time-step t:

$$a_{-i,t} \sim P_{a_{-i,t}}(\cdot|h_{i,t-1}). \tag{23}$$

Assumption (1) suggests that the agents engage in a way that is very similar to a board game like chess, for which player i can make a move only after players $-i$ have made their move. This behaviour is illustrated in Figure 1 (magnified area). Given this assumption, the notation $a_{-i,t}$ can also be seen as referring to actions selected during the interval $(t-\Delta t, t)$.

Assumption 2 (*Exogenous information*). The exogenous information $w_{i,t}^{exog}$ is given by a stochastic model that depends solely on k past values, where $0 < k \leq t$ and a random disturbance $e_{i,t}$ according to:

$$w_{i,t}^{exog} = b(w_{i,t-1}^{exog}, ..., w_{i,t-k}^{exog}, e_t), \tag{24}$$

$$e_{i,t} \sim P_{e_{i,t}}(\cdot|h_{i,t}). \tag{25}$$

Assumption 3 (*Strategy for storage control*). The control decisions related to the charging ($\Delta C_{i,t}(\tau)$) or discharging ($\Delta G_{i,t}(\tau)$) power to/from the storage device are made based on the "default" strategy described in Section 2.3.

It can be observed that with such an assumption, the storage control decisions ($\Delta C_{i,t}(\tau)$ and $\Delta G_{i,t}(\tau)$) are obtained as a direct consequence of the trading decisions $a_{i,t}$. Indeed, after the trading decisions are submitted and the market position is updated, the storage control decisions are subsequently derived following the "default" strategy. Assumption (3) results in reducing the dimensionality of the action space and consequently the complexity of the decision-making problem.

Following Assumptions (1), (2) and (3), one can simply observe that the decision-making problem faced by an agent i operating a storage device and trading energy in the CID market can be formalised as a fully observable finite-time MDP with the following characteristics:

- *Discrete time-step* $t \in T$, where T is the optimisation horizon.
- *State space* H_i, where the state of the system $h_{i,t} \in H_i$ at time t summarises all past information that is relevant for future optimisation.
- *Action space* A_i, where $a_{i,t} \in A_i$ is the set of new orders posted by agent i at time-step t.
- *Transition probabilities* $h_{i,t+1} \sim P(\cdot|h_{i,t}, a_{i,t})$, that can be inferred by the following processes:
  1. $a_{-i,t} \in A_{-i}$ is drawn according to equation (23)
  2. The state of the order book s_{t+1}^{OB} follows the transition given by equation (1)
  3. The exogenous information $w_{i,t}^{exog}$ is given by equation (24) and the noise by (25)
  4. The variable $s_{i,t+1}$ that summarises the information of the storage device optimizing agent follows the transition given by equations (1), (6)-(12) (24), (25) and (16)
  5. The instantaneous reward $r_{i,t}$ collected after each transition is given by equations (18) and (19).

  The elements resulting from these processes can be used to construct $h_{i,t+1}$ in a straightforward way.

3.2. Assumptions on the trading actions

Assumption 4 (*Aggressor*). The trading agent can only submit new orders that match already existing orders at their price (i.e. aggressor or liquidity taker).

Let A_i^{red} be the space that contains only actions that match pre-existing orders in the order book. According to Assumption (4), the i^{th} agent, at time-step t, is restricted to select actions $a_{i,t} \in A_i^{red} \subset A_i$. Let $s_t^{OB} = ((x_j^{OB}, y_j'^{OB}, v_j^{OB}, p_j^{OB}, e_j^{OB}), \forall j \in N_t)$ be the order book observation at trading time-step t. We use y'^{OB} to denote that the new orders have the opposite side ("Buy" or "Sell") than the existing orders. We denote as $a_{i,t}^j \in [0,1]$ the fraction of the volume accepted from order j. The reduced action space A_i^{red} is then defined as:

$$A_i^{red} = \{(x_j^{OB}, y_j'^{OB}, a_{i,t}^j \cdot v_j^{OB}, p_j^{OB}, e_j^{OB}), a_{i,t}^j \in [0,1], \forall j \in N_t\}.$$

At this point, posting a new set of orders $a_{i,t} \in A_i^{red}$ boils down to simply specifying the vector of fractions:

$$\bar{a}_{i,t} = \left(a_{i,t}^j, \forall j \in N_t\right) \in \bar{A}_i^{red}$$

that define the partial or full acceptance of the existing orders. The action $a_{i,t}$ submitted by an aggressor is a function l of the observed order book s_t^{OB} and the vector of fractions $\bar{a}_{i,t}$ and is given by:

$$a_{i,t} = l(s_t^{OB}, \bar{a}_{i,t}). \tag{26}$$

3.3. Restrictions on the storage operation

Assumption 5 (*No imbalances permitted*). The trading agent can only accept an order to buy or sell energy if and only if it does not result in any imbalance for the remaining delivery periods.

According to Assumption (5) the agent is completely risk-averse in the sense that, even if it stops trading at any given point, its position in the market can be covered without causing any imbalance. This assumption is quite restrictive with respect to the full potential of an asset-optimizing agent in the CID market. We note that, according to the German regulation policies (see (Braun & Hoffmann, 2016)), the imbalance market should not be considered as an optimisation floor and the storage device should always be balanced at each trading time-step t ($\Delta_{i,t}(\tau) = 0, \forall \tau \in \bar{T}$). In this respect, we can view Assumption 5 as a way to comply with the German regulation policies in a risk-free context where each new trade should not create an imbalance that would have to be covered later.

Assumption 6 (*Optimization decoupling*). The storage device has a given initial value for the storage level SoC_i^{init} at the beginning of the delivery timeline. Moreover, it is constrained to terminate at a given level SoC_i^{term} at the end of the delivery timeline.

Under Assumption (6) the optimisation of the storage unit over a long trading horizon can be decomposed into shorter optimisation windows (e.g. of one day). In the simulation results reported later in this paper, we will choose $SoC_i^{init} = SoC_i^{term}$.

4. Methodology

In this section, we describe the methodology that has been applied for tackling the MDP problem described in subsection 3. We consider that, in reality, an asset-optimizing agent has at its disposal a set of trajectories (one per day) from participating in the CID market in the past years. The process of collecting these trajectories and their structure is presented in Section 4.1. Based on this dataset, we propose in subsection 4.2 the deployment of the fitted Q iteration algorithm as introduced in (Ernst et al., 2005). This algorithm belongs to the class of batch-mode RL algorithms that make use of all the available samples at once for updating the policy. This class of algorithms is known to be very sample efficient.

Despite the different assumptions made on the operation of the storage device and the way it is restricted to interact with the market, the dimensionality of the action space still remains very high. Due to limitations related to the function approximation architecture used to implement the fitted Q iteration algorithm, a low-dimensional and discrete action space is necessary, as discussed in subsection 4.3. Therefore, as part of the methodology, in subsection 4.4 we propose a way for reducing the action space. Afterwards, in subsection 4.5, a more compact representation of the state space is proposed in order to reduce the computational complexity of the training process and increase the sample efficiency of the algorithm.

Finally, the low number of available samples (one trajectory per day) gives rise to issues related to the limited exploration of the agent. In order to address these issues, we generate a large number of trading trajectories of our MDP according to an ε-greedy policy, using historical trading data. In the last part of this section, we elaborate on the strategy that is used in this paper for generating the trajectories and the limitations of this procedure.

4.1. Collection of trajectories

As previously mentioned, an asset-optimizing agent can collect a set of trajectories from previous interactions with the CID market. Based on Assumption (6), each day can be optimised separately and thus, trading for one day corresponds to one trajectory. We consider that the trading horizon defined in Section 2.2 consists of K discrete trading time-steps such that $T = \{0, ..., K\}$. A single trajectory sampled from the MDP described in Section 3 is defined as:

$$\zeta_m = \left(h_{i,0}^m, a_{i,0}^m, r_{i,0}^m, ..., h_{i,K-1}^m, a_{i,K-1}^m, r_{i,K-1}^m, h_{i,K}^m\right).$$

A set of M trajectories can be then defined as:

$$F = \{\zeta_m, m = 1, ..., M\}.$$

The set of trajectories F can be used to generate the set of sampled one-step system transitions F' defined as:

$$F' = \begin{Bmatrix} (h_{i,0}^1, a_{i,0}^1, r_{i,0}^1, h_{i,1}^1), & \cdots & (h_{i,K-1}^1, a_{i,K-1}^1, r_{i,K-1}^1, h_{i,K}^1), \\ \vdots & \ddots & \vdots \\ (h_{i,0}^M, a_{i,0}^M, r_{i,0}^M, h_{i,1}^M), & \cdots & (h_{i,K-1}^M, a_{i,K-1}^M, r_{i,K-1}^M, h_{i,K}^M) \end{Bmatrix}.$$

The set F' is split into K sets of one-step system transitions F_t' defined as:

$$F_t' = \left\{(h_{i,t}^m, a_{i,t}^m, r_{i,t}^m, h_{i,t+1}^m), m = 1, ..., M\right\}_t,$$
$$\forall t \in \{0, ..., K-1\}.$$

A batch-mode RL algorithm can be implemented for extracting a relevant trading policy from these one-step system transitions. In the following subsection, the type of RL algorithm used for inferring a high-quality policy from this set of one-step system transitions is explained in detail.

4.2. Batch-mode reinforcement learning

Q-functions and Dynamic Programming: In this section, the fitted Q iteration algorithm is proposed for the optimisation of the MDP defined in Section 3, using a set of collected trajectories. In order to solve the problem, we first define the Q-function for each state-action pair $(h_{i,t}, a_{i,t})$ at time t as proposed in (Bertsekas, 2005) as:

$$Q_t(h_{i,t}, a_{i,t}) = \underset{a_{-i,t},\, e_{i,t}}{\mathbb{E}} \{r_{i,t} + V_{t+1}(h_{i,t+1})\}, \quad \forall t \in \{0, ..., K-1\}. \tag{27}$$

A time-variant policy $\pi = \{\mu_0, ..., \mu_{K-1}\} \in \Pi$, consists in a sequence of functions μ_t, where $\mu_t : H_i \rightarrow A_i^{red}$. An action $a_{i,t}$ is selected from this policy at each time-step t, according to $a_{i,t} = \mu_t(h_{i,t})$. We denote as $\pi^{t+1} = \{\mu_{t+1}, ..., \mu_{K-1}\}$ the sequence of functions μ_t from time-step $t+1$ until the end of the horizon. Standard results from dynamic programming (DP) show that for the finite time MDP we are addressing in this paper, there exists at least one such time-variant policy which is an optimal policy as defined by equation (22). Therefore, we focus on the computation of such an optimal time-variant policy. We define the value function V_{t+1} as the optimal expected cumulative rewards from stage $t+1$ until the end of the horizon K given by:

$$V_{t+1}(h_i) = \max_{\pi^{t+1} \in \Pi} \underset{\substack{(a_{-i,t+1}, e_{i,t+1}) \\ \cdots \\ (a_{-i,K-1}, e_{i,K-1})}}{\mathbb{E}} \left\{ \sum_{k=t+1}^{K-1} R_{i,k}\left(h_{i,k}, \mu_k(h_{i,k}), a_{-i,k}\right) | h_{i,t+1} = h_i \right\}. \tag{28}$$

We observe that $Q_t(h_{i,t}, a_{i,t})$ is the value attained by taking action $a_{i,t}$ at state $h_{i,t}$ and subsequently using an optimal policy. Using the dynamic programming algorithm (Bertsekas, 2005) we have:

$$V_t(h_{i,t}) = \max_{a_{i,t} \in A_i^{red}} Q_t(h_{i,t}, a_{i,t}). \tag{29}$$

Equation (27) can be written in the following form that relates Q_t and Q_{t+1}:

$$Q_t(h_{i,t}, a_{i,t}) = \underset{a_{-i,t},\, e_{i,t}}{\mathbb{E}} \left\{ r_{i,t} + \max_{a_{i,t+1} \in A_i^{red}} Q_{t+1}(h_{i,t+1}, a_{i,t+1}) \right\}. \tag{30}$$

An optimal time-variant policy $\pi^* = \{\mu_0^*, ..., \mu_{K-1}^*\}$ can be identified using the Q-functions as following:

$$\mu_t^* = \arg\max_{a_{i,t} \in A_i^{red}} Q_t(h_{i,t}, a_{i,t}), \quad \forall t \in \{0, ..., K-1\}. \tag{31}$$

Computing the Q-functions from a set of one-step system transitions: In order to obtain the optimal time-variant policy π^*, the effort is focused on computing the Q-functions defined in equation (30). However, two aspects render the use of the standard value iteration algorithm impossible for solving the MDP defined in Section 3. First, the transition probabilities of the MDP defined in Section 3 are not known. Instead, we can exploit the set of collected historical trajectories to compute the exact Q-functions using an algorithm such as Q-learning (presented in (Watkins & Dayan, 1992)). Q-learning is designed for working only with trajectories, without any knowledge of the transition probabilities. Optimality is guaranteed given that all state-action pairs are observed infinitely often within the set of the historical trajectories and that the successor states are independently sampled at each occurrence of a state-action pair (Bertsekas, 2005). In Section 4.6 we discuss the validity of this condition and we address the problem of limited exploration by generating additional artificial trajectories. Second, due to the continuous nature of the state and action spaces a tabular representation of the Q-functions used in Q-learning is not feasible. In order to overcome this issue, we use a function approximation architecture to represent the Q-functions (Busoniu et al., 2017).

The computation of the approximate Q-functions is performed using the fitted Q iteration algorithm (Ernst et al., 2005). We present the algorithm for the case where a parametric function approximation architecture ($Q_t(h_{i,t}, a_t; \theta_t)$) is used (e.g. neural networks). In this case, the algorithm is used to compute, recursively, the parameter vectors θ_t starting from $t = K-1$. However, it should be emphasized that the fitted Q iteration algorithm can be adapted in a straightforward way to the case in which a non-parametric function approximation architecture is selected.

The set of M samples of quadruples $F_t' = \left\{(h_{i,t}^m, a_{i,t}^m, r_t^m, h_{i,t+1}^m), m = 1, ..., M\right\}$ obtained from previous experience is exploited in order to update the parameter vectors θ_t by solving the supervised learning problem presented in equation (32). The target vectors y_t are

computed using the Q-function approximation of the next stage ($Q_{t+1}(h_{i,t+1}, a_{t+1}; \theta_{t+1})$) according to equation (33). The Q-function for the terminal state is set to zero ($\hat{Q}_K \equiv 0$) and the algorithm iterates backwards in the time horizon T, producing a sequence of approximate Q-functions denoted by $\hat{Q} = \{\hat{Q}_0, ..., \hat{Q}_{K-1}\}$ until termination at $t = 0$.

$$\theta_t = \arg\min_{\theta_t} \sum_{m=1}^{M} (Q_t(h_{i,t}^m, a_t^m; \theta_t) - y_t^m)^2 \tag{32}$$

$$y_t^m = r_t^m + \max_{a_{i,t+1} \in A_i^{red}} Q_{t+1}(h_{i,t+1}^m, a_{i,t+1}; \theta_{t+1}) \tag{33}$$

Once the parameters θ_t are computed, the time-variant policy $\hat{\pi}^* = \{\hat{\mu}_0^*, ..., \hat{\mu}_{K-1}^*\}$ is obtained as:

$$\hat{\mu}_t^*(h_{i,t}) = \arg\max_{a_{i,t} \in A_i^{red}} Q_t(h_{i,t}, a_{i,t}; \theta_t), \quad \forall t \in \{0, ..., K-1\}. \tag{34}$$

In practice, a new trajectory is collected after each trading day. The set of collected trajectories F is consequently augmented. Thus, the fitted Q iteration algorithm can be used to compute a new optimal policy when new data arrive.

4.3. Limitations

The fitted Q iteration algorithm, described in the previous section, can be used to provide a trading policy based on the set of past trajectories at the disposal of the agent. Even though, this approach is theoretically sound, in practice there are several limitations to overcome. The efficiency of the described fitted Q iteration algorithm is overshadowed by the high-dimensionality of the state and the action space.

The state variable

$$h_{i,t} = (s_{i,0}, a_{i,0}, r_{i,0}, ..., s_{i,t-1}, a_{i,t-1}, r_{i,t-1}, s_{i,t}) \in H_i$$

is composed of :

- The entire history of actions $(a_{i,0}, ..., a_{i,t-1})$ before time t
- The entire history of rewards $(r_{i,0}, ..., r_{i,t-1})$ before time t
- The history of order book states $\left(s_0^{OB}, ..., s_t^{OB}\right)$ up to time t and,y of the private information $(s_{i,0}^{private}, ..., s_{i,t}^{private})$ up to time t, where:

$$s_{i,t}^{private} = ((P_{i,t}^{mar}(\tau), \Delta_{i,t}(\tau), G_{i,t}(\tau), C_{i,t}(\tau), SoC_{i,t}(\tau), \forall \tau \in \bar{T}), w_{i,t}^{exog}).$$

The state space H_i as well as the action space A_i^{red}, as described in Section 3.2, depend explicitly on the content of the order book s_t^{OB}. The dimension of these spaces at each time-step t depends on the total number of available orders $| N_t |$ in the order book. However, the total number of orders is changing at each step t. Thus, both the state and the action spaces are high-dimensional spaces of variable size. In order to reduce the complexity of the decision-making problem, we have chosen to reduce these spaces so as to work with a small action space of constant size and a compact state space. In the following, we describe the procedure that was carried out for the reduction of the state and action spaces.

4.4. Action space reduction: High-level actions

In this section, we elaborate on the design of a small and discrete set of actions that is an approximation of the original action space. Based on Assumptions (1), (2), (3), (4), (5) and (6), a new action space A_i' is proposed, which is defined as $A_i' = \{\text{“}Trade\text{”}, \text{“}Idle\text{”}\}$. The new action space is composed of two high-level actions $a_{i,t}' \in A_i'$. These high-level actions are transformed to an original action through mapping $p : A_i' \to A_i^{red}$, from space A_i' to the reduced action space A_i^{red}. The high-level actions are defined as following:

4.4.1. “TRADE”

At each time-step t, agent i selects orders from the order book with the objective of maximizing the instantaneous reward under the constraint that the storage device can remain balanced for every delivery period, even if no further interaction with the CID market occurs. As a reminder, this constraint was imposed by Assumption (5).

Under this assumption, the instantaneous reward signal $r_{i,t}$, presented in equation (19), consists only of the trading revenues obtained from the matching process of orders at time-step t. We will further assume that mapping $u : \mathbb{R}^+ \times \{\text{“}Sell\text{”}, \text{“}Buy\text{”}\} \to \mathbb{R}$ that adjusts the sign of the volume v^{OB} of each order according to their side y^{OB}. Orders posted for buying energy will be associated with positive volume and orders posted for selling energy with negative volume, or equivalently:

$$u(v^{OB}, y^{OB}) = \begin{cases} v^{OB}, & \text{if } y^{OB} = \text{“}Buy\text{”}, \\ -v^{OB}, & \text{if } y^{OB} = \text{“}Sell\text{”}. \end{cases} \tag{35}$$

Consequently, the reward function ρ defined in Section 2.4 is adapted according to the proposed modifications. The new reward function ρ, where $\rho : S^{OB} \times \bar{A}_i^{red} \to \mathbb{R}$, is a stationary function of the orders observed at each time-step t and the agent's response to the observed orders. An analytical expression for the instantaneous reward collected

is given by:

$$r_{i,t} = \rho\left(s_t^{OB}, \bar{a}_{i,t}\right) = \sum_{j=1}^{N_t} a_{i,t}^j \cdot u(v_j^{OB}, y_j^{OB}) \cdot p_j^{OB}. \quad (36)$$

The High-level action "Trade" amounts to solving the bid acceptance optimisation problem presented in Algorithm 1. The objective function of the problem, formulated in equation (37), consists of the revenues arising from trading. It is important to note that the operational constraints guarantee that no order will be accepted if it causes any imbalance. We denote as $N_\tau \subset \mathbb{N}$ the set of unique indices of the available orders that correspond to delivery time-step τ and $N_t = \bigcup_{\tau \in \bar{T}} N_\tau$. In equation (38), the energy purchased and sold ($\sum_{j \in N_\tau} a_{i,t}^j u(v_j^{OB})$), the past net energy trades ($P_{i,t}^{mar}(\tau)$) and the energy discharged by the storage ($G_{i,t}(\tau)$) must match the energy charged by the storage ($C_{i,t}(\tau)$) for every delivery time-step τ. The energy balance of the storage device, presented in equation (39), is responsible for the time-coupling and the arbitrage between two products x (delivery time-steps τ). The technical limits of the storage level and the charging and discharging process are described in equations (40) to (44). The binary variables $k_{i,t} = (k_{i,t}(\tau), \forall \tau \in \bar{T})$ restrict the operation of the unit for each delivery period in only one mode, either charging or discharging.

The optimal solution to this problem yields the vector of fractions:

$$\bar{a}_{i,t} = \left(a_{i,t}^j, \forall j \in N_t\right) \in \bar{A}_i^{red}$$

that are used in equation (26) to construct the action $a_{i,t} \in A_i^{red}$. The optimal solution also defines at each time-step t the adjustments in the level of the production (discharge) $\Delta G_{i,t} = (\Delta G_{i,t}(\tau), \forall \tau \in \bar{T}(t))$ and the consumption (charge) $\Delta C_{i,t} = (\Delta C_{i,t}(\tau), \forall \tau \in \bar{T}(t))$. The evolution of the state of charge $SoC_{i,t+1} = (SoC_{i,t+1}(\tau), \forall \tau \in \bar{T}(t))$ of the unit as well as the production $G_{i,t+1} = (G_{i,t+1}(\tau), \forall \tau \in \bar{T}(t))$ and consumption $C_{i,t+1} = (C_{i,t+1}(\tau), \forall \tau \in \bar{T}(t))$ levels are computed for each delivery period.

4.4.2. "IDLE"

No transactions are executed, and no adjustment is made to the previously scheduled quantities. Under this action, the vector of fractions $\bar{a}_{i,t}$ is a zero vector. The discharge and charge as well as the state of charge of the storage device remain unchanged ($\Delta G_{i,t} \equiv 0$ and $\Delta C_{i,t} \equiv 0$) and we have:

$$G_{i,t+1}(\tau) = G_{i,t}(\tau), \forall \tau \in \bar{T}(t), \quad (49)$$
$$C_{i,t+1}(\tau) = C_{i,t}(\tau), \forall \tau \in \bar{T}(t), \quad (50)$$
$$SoC_{i,t+1}(\tau) = SoC_{i,t}(\tau), \forall \tau \in \bar{T}(t). \quad (51)$$

With such a reduction of the action-space, the agent can choose at every time-step t between the two described high-level actions ($a'_{i,t} \in A'_i = \{$"$Trade$","$Idle$"$\}$). Note that

Algorithm 1 "Trade"

Input: t, s_t^{OB}, $P_{i,t}^{mar}$, $\underline{SoC_i}$, $\overline{SoC_i}$, $\underline{C_i}$, $\overline{C_i}$, $\underline{G_i}$, $\overline{G_i}$, SoC_i^{init}, SoC_i^{term}, τ_{init}, τ_{term}, $G_{i,t}$, $C_{i,t}$

Output: $\bar{a}_{i,t}$, $SoC_{i,t+1}$, $G_{i,t+1}$, $C_{i,t+1}$, $\Delta G_{i,t}$, $\Delta C_{i,t}$, $k_{i,t+1}$, $r_{i,t}$

Solve:

$$\max_{\substack{\bar{a}_{i,t},\, SoC_{i,t+1} \\ G_{i,t+1},\, C_{i,t+1} \\ \Delta G_{i,t},\, \Delta C_{i,t} \\ k_{i,t+1},\, r_{i,t}}} \sum_{j \in N_t} a_{i,t}^j \cdot u(v_j^{OB}, y_j^{OB}) \cdot p_j^{OB} \quad (37)$$

s.t. $\sum_{j \in N_\tau} a_{i,t}^j u(v_j^{OB}, y_j^{OB}) + P_{i,t}^{mar}(\tau) +$

$$C_{i,t+1}(\tau) = G_{i,t+1}(\tau), \quad \forall \tau \in \bar{T}(t) \quad (38)$$
$$SoC_{i,t+1}(\tau + \Delta\tau) = SoC_{i,t+1}(\tau) +$$
$$\Delta\tau \cdot \left(\eta \cdot C_{i,t+1}(\tau) - \frac{G_{i,t+1}(\tau)}{\eta}\right), \quad \forall \tau \in \bar{T}(t) \quad (39)$$
$$\underline{SoC_i} \le SoC_{i,t+1}(\tau) \le \overline{SoC_i}, \quad \forall \tau \in \bar{T}(t) \quad (40)$$
$$SoC_i^{init} = SoC_{i,t+1}(\tau_{init}), \quad (41)$$
$$SoC_i^{term} = SoC_{i,t+1}(\tau_{term}), \quad (42)$$
$$\underline{C_i} \le C_{i,t+1}(\tau) \le k_{i,t+1}(\tau) \cdot \overline{C_i}, \quad \forall \tau \in \bar{T}(t) \quad (43)$$
$$\underline{G_i} \le G_{i,t+1}(\tau) \le (1 - k_{i,t+1}(\tau))\,\overline{G_i}, \quad \forall \tau \in \bar{T}(t) \quad (44)$$
$$G_{i,t+1}(\tau) = G_{i,t}(\tau) + \Delta G_{i,t}(\tau), \quad \forall \tau \in \bar{T}(t) \quad (45)$$
$$C_{i,t+1}(\tau) = C_{i,t}(\tau) + \Delta C_{i,t}(\tau), \quad \forall \tau \in \bar{T}(t) \quad (46)$$
$$k_{i,t+1}(\tau) \in \{0, 1\}, \quad \forall \tau \in \bar{T}(t) \quad (47)$$
$$a_{i,t}^j \in [0, 1], \quad \forall j \in N_t \quad (48)$$

when the agent learns to idle, given a current situation, it does not necessarily mean, that if it had chosen to "*Trade*" instead, he would not make a positive immediate reward. Indeed, the agent would choose "*Idle*" if it believes that there may be a better market state emerging, i.e. the agent would learn to wait for the ideal opportunity of orders appearing in the order book at subsequent time-steps. We compare this approach to an alternative, which we refer to as the "rolling intrinsic" policy. According to this policy, at every time-step t of the trading horizon the agent selects the combination of orders that optimises its operation and profits, based on the current information assuming that the storage device must remain balanced for every delivery period as presented in (Lohndorf & Wozabal, 2015). The "rolling intrinsic" policy is, thus, equivalent to sequentially selecting the action "*Trade*" (Algorithm 1), as defined in this framework. The algorithm proposed later in this paper exploits the experience that the agent can gain through (artificial) interaction with its environment, in order to learn the value of trading or idling at every different state that agent may encounter.

4.5. State space reduction

In this section, we propose a more compact and low-dimensional representation of the state space H_i. The state $h_{i,t}$, as explained in Section 4.3, contains the entire history of all the relevant information available for the decision-making up to time t. We consider each one of the components of the state $h_{i,t}$, namely the entire history of actions, order book states and private information, and we provide a simplified form.

First, the vector containing the entire history of actions is reduced to a vector of binary variables after the modifications introduced in Section 4.4.

Second, the vector containing the history of order book states is reduced into a vector of engineered features. We start from the order book state $s_t^{OB} = ((x_j^{OB}, y_j'^{OB}, v_j^{OB}, p_j^{OB}, e_j^{OB}), \forall j \in N_t \subseteq \mathbb{N}) \in S^{OB}$ that is defined in Sections 2.1 and 2.2 as a high-dimensional continuous vector used to describe the state of the CID market. Owing to the variable (non-constant) and large amount of orders $|N_t|$, the space S^{OB} has a non-constant size with high-dimensionality.

In order to overcome this issue, we proceed as following. First, we consider the market depth curves for each product x. The market depth of each side ("Sell" or "Buy") at a time-step t, is defined as the total volume available in the order book per price level for product x. The market depth for the "Sell" ("Buy") side is computed by stacking the existing orders in ascending (descending) price order and accumulating the available volume. The market depth for each of the quarter-hourly products Q_1 to Q_6 at time instant t is illustrated in Figure 2a using data from the German CID market. The market depth curves serve as a visualization of the order book that provides information about the liquidity of the market. Moreover, it provides information about the maximum (minimum) price that a trading agent will have to pay in order to buy (sell) a certain volume of energy. If we assume a fixed-price discretisation, certain upper and lower bounds on the prices and interpolation of the data in this price range, the market depth curves of each product x can be approximated by a finite and constant set of values.

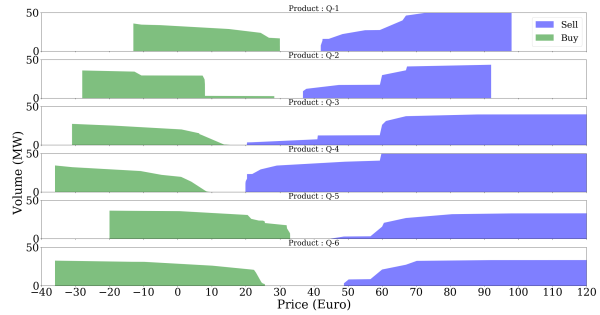
Even though this set of values has a constant size, it can still be extremely large. Its dimension is not a function of the number of existing orders any more, but it depends on the resolution of the price discretisation, the price range considered, and the total number of products in the market. Instead of an individual market depth curve for each product x, we consider a market depth curve for all the available products, i.e. existing orders in ascending (descending) price order and accumulating the available volumes for all the products. In this way we can construct the aggregated market depth curve, presented in Figure 2b. The aggregated market depth curve illustrates the total available volume ("Sell" or "Buy") per price level for all products.

The motivation for considering the aggregated curves comes from the very nature of a storage device. The main profit-generating mechanism of a storage device is the arbitrage between two delivery periods. Its functionality involves the purchasing (charging) of electricity during periods of low prices and the selling (discharging) during periods of high prices.

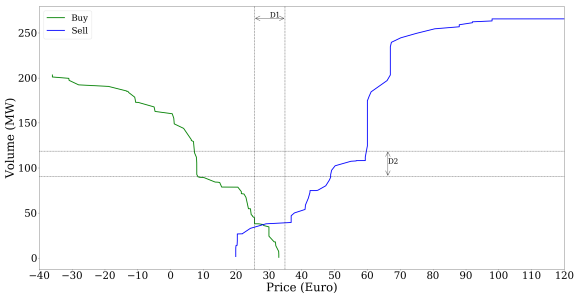
For instance, in Figure 2a, a storage device would buy volume for product Q_4 and sell volume back for product Q_5. The intersection of the "Sell" and "Buy" curves in Figure 2b defines the maximum volume that can be arbitraged by the storage device if no operational constraints were considered and serves as an upper bound for the profits at each step t. Alternatively, the market depth for the same products Q_1 to Q_6 at a different time-step of the trading horizon is presented in Figure 3a. As illustrated in Figure 3b, there is no arbitrage opportunity between the products, hence the aggregated curves do not intersect. Thus, we assume, that the aggregated curves provide a sufficient representation of the order book.

At this point, considering a fixed-price discretisation and a fixed price range would yield a constant set of values able to describe the aggregated curves. However, in order to further decrease the size of the set of values with sufficient price discretisation, we motivate the use of a set of distance measures between the two aggregated curves that succeed in capturing the arbitrage potential at each trading time-step t as state variables, as presented in Figures 2b and 3b.

For instance, we define as $D1$ the signed distance between

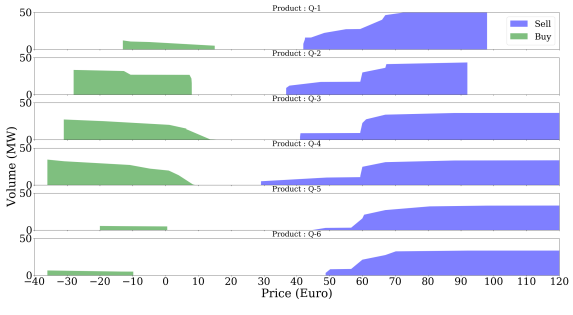


(a)

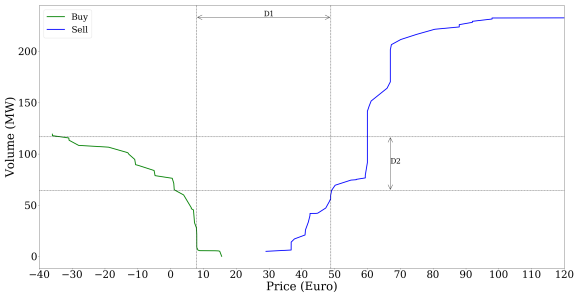


(b)

Figure 2. (a) Market depth per product (for products Q_1 to Q_6) at a time-step t with arbitrage potential. (b) The corresponding aggregated curves for a profitable order book.



(a)



(b)

Figure 3. (a) Market depth per product (for products Q_1 to Q_6) at a time-step t with no arbitrage potential. (b) The corresponding aggregated curves for a non profitable order book.

the 75th percentile of "Buy" price and the 25th percentile of "Sell" price and as $D2$ the absolute distance between the mean value of "Buy" and "Sell" volumes. Other measures used are the signed price difference and absolute volume difference between percentiles (25%, 50%, 75%) and the bid-ask spread. A detailed list of the distance measures is provided in Table 2.

The new, continuous, low-dimensional observation of the order book $s_t'^{OB} \in S'^{OB} = \{D1, .., D10\}$ is used to represent the state of the order book and, in particular, its profit potential. It is important to note that in contrast to $s_t^{OB} \in S^{OB}$, the new order book observation $s_t'^{OB} \in S'^{OB}$ does not depend on the number of orders in the order book and therefore has a constant size, i.e. the cardinality of S'^{OB} is constant over time.

Finally, the history of the private information of agent i, that is not publicly available, is a vector that contains the high-dimensional continuous variables $s_{i,t}^{private}$ related to the operation of the storage device. As described in Section 4.3, $s_{i,t}^{private}$ is defined as:

$$\begin{aligned} s_{i,t}^{private} = ((&P_{i,t}^{mar}(\tau), \Delta_{i,t}(\tau), \\ &G_{i,t}(\tau), C_{i,t}(\tau), SoC_{i,t}(\tau), \forall \tau \in \bar{T}), \\ &w_{i,t}^{exog}). \end{aligned}$$

According to Assumption (5), the trading agent cannot perform any transaction if it results in imbalances. Therefore, it is not relevant to consider the vector $\Delta_{i,t}$ since it will always be zero according to the way the high-level actions are defined in Section 4.4. Additionally, Assumption (3) regarding the default strategy for storage control in combination with Assumption (5) yields a direct correlation between vectors $P_{i,t}^{mar}$ and $G_{i,t}$, $C_{i,t}$, $SoC_{i,t}$. Thus, it is considered that $P_{i,t}^{mar}$ contains all the required information and thus vectors $G_{i,t}$, $C_{i,t}$ and $SoC_{i,t}$ can be dropped.

Following the previous analysis we can define the low-dimensional pseudo-state $z_{i,t} = (s'_{i,0}, a'_{i,0}, r_{i,0}, ..., a'_{i,t-1}, r_{i,t-1}, s'_{i,t}) \in Z_i$, where $s'_{i,t} = (s_t'^{OB}, P_{i,t}^{mar}, w_{i,t}^{exog}) \in S'_i$. This pseudo-state can be seen as the result of applying an encoder $e : H_i \to Z_i$ which maps a true state $h_{i,t}$ to pseudo-state $z_{i,t}$.

Assumption 7 (*Pseudo-state*). The pseudo-state $z_{i,t} \in Z_i$ contains all the relevant information for the optimisation of the CID market trading of an asset-optimizing agent.

Under Assumption (7), using the pseudo-state $z_{i,t}$ instead of the true state $h_{i,t}$ is equivalent and does not lead to a sub-optimal policy. The resulting decision process after the state and action spaces reductions is illustrated in Figure 4.

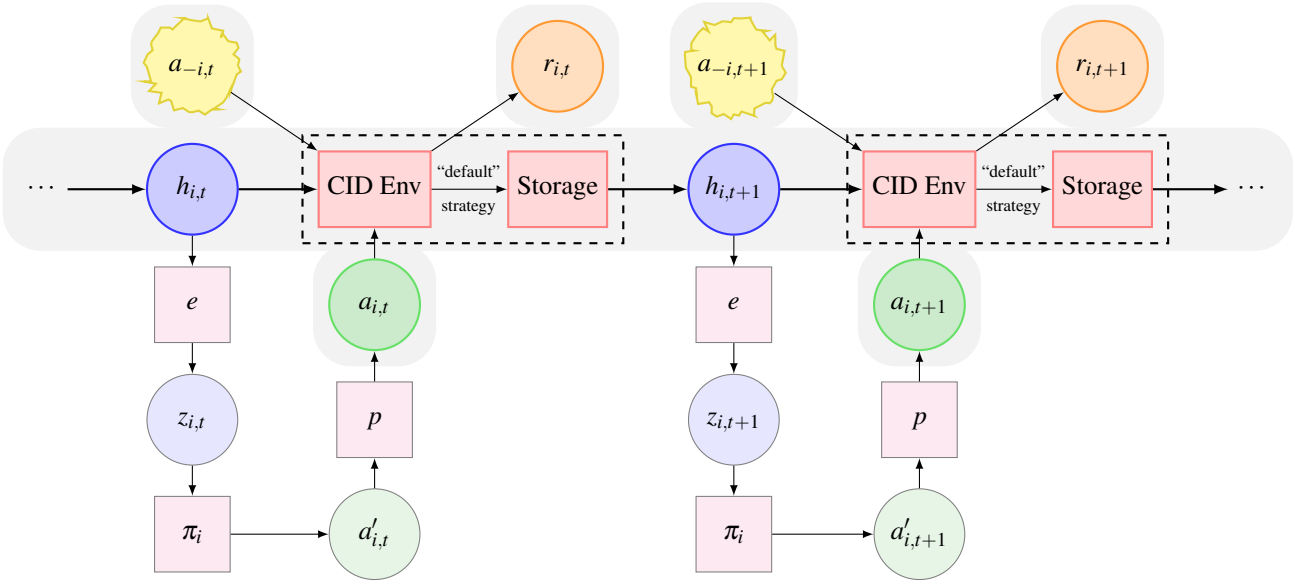


Figure 4. Schematic of the decision process. The original MDP is highlighted in a gray background. The state of the original MDP $h_{i,t}$ is encoded in pseudo-state $z_{i,t}$. Based on $z_{i,t}$, agent i takes an high-level action $a'_{i,t}$, according to its policy π_i. This action $a'_{i,t}$ is mapped to an original action $a_{i,t}$ and submitted to the CID market. The CID market makes a transition based on the action of agent i and the actions of the other agents $a_{-i,t}$. After this transition, the market position of agent i is defined and the control actions for storage device are derived according to the "default" strategy. Each transition yields a reward $r_{i,t}$ and a new state $h_{i,t}$.

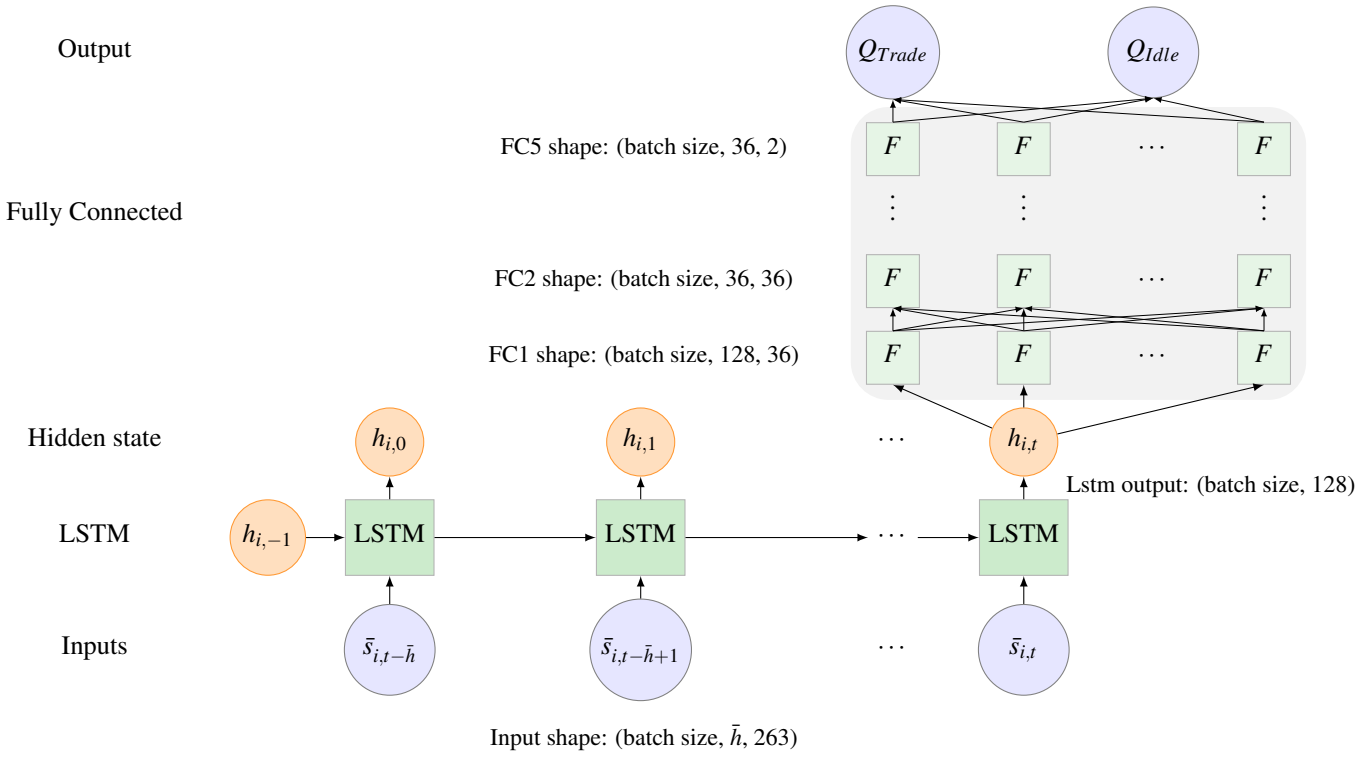


Figure 5. Schematic of the neural network architecture.

Table 2. Order book features used for the state reduction.

Symbol	Definition	Description
$D1$	$p^{Buy}_{max} - p^{Sell}_{min}$	Signed diff. between the maximum "Buy" price and the minimum "Sell" price
$D2$	$p^{Buy}_{mean} - p^{Sell}_{mean}$	Signed diff. between the mean "Buy" price and the mean "Sell" price
$D3$	$p^{Buy}_{25\%} - p^{Sell}_{75\%}$	Signed diff. between the 25th percentile "Buy" price and the 75th percentile "Sell" price
$D4$	$p^{Buy}_{50\%} - p^{Sell}_{50\%}$	Signed diff. between the 50th percentile "Buy" price and the 50th percentile "Sell" price
$D5$	$p^{Buy}_{75\%} - p^{Sell}_{25\%}$	Signed diff. between the 75th percentile "Buy" price and the 25th percentile "Sell" price
$D6$	$\lvert v^{Buy}_{min} - v^{Sell}_{min} \rvert$	Abs. diff. between the minimum "Buy" cum. volume and the maximum "Sell" cum. volume
$D7$	$\lvert v^{Buy}_{mean} - v^{Sell}_{mean} \rvert$	Abs. diff. between the mean "Buy" cum. volume and the mean "Sell" cum. volume
$D8$	$\lvert v^{Buy}_{25\%} - v^{Sell}_{25\%} \rvert$	Abs. diff. between the 25th percentile "Buy" cum. volume and the 25th percentile "Sell" cum. volume
$D9$	$\lvert v^{Buy}_{50\%} - v^{Sell}_{50\%} \rvert$	Abs. diff. between the 50th percentile "Buy" cum. volume and the 50th percentile "Sell" cum. volume
$D10$	$\lvert v^{Buy}_{75\%} - v^{Sell}_{75\%} \rvert$	Abs. diff. between the 75th percentile "Buy" cum. volume and the 75th percentile "Sell" cum. volume

4.6. Generation of artificial trajectories

In this section, the generation of additional artificial trajectories for addressing exploration issues is discussed. Indeed, if we were to implement an agent that selects at every timestep among the "Idle" and "Trade" actions, we would collect a certain number of trajectories (one per day) over a certain period of interactions. The collected dataset could be used to train a policy using a batch mode RL algorithm, as described in Section 4.2. Every time a new trajectory would arrive, it would be appended in the previous set of trajectories and the entire dataset could be used to improve the trading policy.

This approach requires a large number of days in order to acquire a sufficient amount of data. Additionally, as discussed in Section 4.2, sufficient exploration of the state and action spaces is a key requirement for converging to a near-optimal policy. It is required that all parts of the state and action spaces are visited sufficiently often. As a result, the RL agent needs to explore unknown grounds in order to discover interesting policies (exploration). It should also apply these learned policies to get high rewards (exploitation). However, since the set of collected trajectories would come from a real agent, the visitation of many different states is expected to be limited.

In the RL context, exploration is performed when the agent selects a different action than the one that, according to its experience, will yield the highest rewards. In real life, it is unlikely for a trader to select such actions, and potentially bear negative revenues, for the sake of gaining more experience. This leads to limited exploration of the learning process and would result in a suboptimal policy.

Assumption 8 (*No impact on the behaviour of the rest of the agents*). The actions of trading agent i do not influence the future actions of the rest of the agents $-i$ in the CID market. In this way, agent i is not capable of influencing the market.

Assumption (8) implies that each of the agents $-i$ entering in the market would post orders solely based on their individual needs. Furthermore, its actions are not considered as a reaction to the actions of the other market players.

Leveraging Assumption (8) allows one to tackle the exploration issues discussed previously by generating several artificial trajectories using historical order book data. We denote by E the number of episodes (times) each day from historical data is repeated and by L^{train} the set of trading days used to train the agent. We can then obtain the total number of trajectories M as $M = E \cdot |L^{train}|$.

The generation of artificial trajectories is performed according to the process described in (Ernst et al., 2005). This process interleaves the generation of trajectories with the computation of an approximate Q-function using the trajectories already generated. As shown in Algorithm 2, for a number of episodes ep, we randomly select days from the training set which we simulate using an ε-greedy policy. According to this policy, an action is chosen at random with probability ε and according to the available Q-functions with probability $(1-\varepsilon)$. The generated trajectories are added to the set of trajectories. The second step consists of updating the Q-function approximation using the set of collected trajectories. This process is terminated when the total number of episodes has reached the specified number E.

This process introduces parameters L^{train}, E, ep, ε and $decay$. The selection of these parameters impacts the training progress and the quality of the resulting policy. The set

Algorithm 2 Generation of artificial trajectories

Input: L^{train}, E, ep, ε, $decay$
Output: $\hat{Q}$, F
Initialize $\hat{Q} \equiv 0$
$M \leftarrow E \cdot |L^{train}|$
$m \leftarrow 0$
repeat
  for $iter_j \leftarrow 0$ **to** ep **do**
    $d \leftarrow rand(L^{train})$ ▷ Randomly pick a day d from train set L^{train}
    $\zeta_m \leftarrow simulate(d, \varepsilon - greedy(\hat{Q}))$ ▷ Generate trajectory ζ_m by simulating day d using ε-greedy policy
    $F.add(\zeta_m)$ ▷ Append trajectory from day d to set F
    $\varepsilon \leftarrow anneal(\varepsilon, decay, iter_i)$ ▷ Anneal the value of ε based on $decay$ parameter
    $m \leftarrow m + 1$
  Update $\hat{Q}$ using set F according to equations (32), (33) ▷ Fit new $\hat{Q}$ functions
until $m \geq M$;
return $\hat{Q}, F$

of days considered for training (L^{train}) is typically selected as a proportion (e.g. 70%) of the total set of days available. The total number of episodes E should be large enough so that convergence is achieved and is typically tuned based on the application. The frequency with which the trajectory generation and the updates are interleaved is controlled by parameter ep. A small number of ep results in a large number of updates. Parameter ε is used to address the trade-off between exploration-exploitation during the training process. As the training evolves, this parameter is annealed based on some predefined parameter $decay$, in order to gradually reduce exploration and to favour exploration along the (near-)optimal trajectories. In practice, the size of the buffer F cannot grow infinitely due to memory limitations, so typically a limit on the number of trajectories stored in the buffer is imposed. Once this limit is reached, the oldest trajectories are removed as new ones arrive. The buffer is a double-ended queue of fixed size.

4.7. Neural Network architecture

As described in Section 4.5, pseudo-state $z_{i,t}$ contains a sequence of variables whose length is proportional to t. This motivates the use of Recurrent Neural Networks (RNNs), that are known for being able to efficiently process variable-length sequences of inputs. In particular, we use Long Short-term Memory (LSTM) networks (Goodfellow et al., 2016), a type of RNNs where a gating mechanism is introduced to regulate the flow of information to the memory state.

All the networks in this study have the architecture presented in Figure 5. It is composed of one LSTM layer with 128 neurons followed by five fully connected layers with 36 neurons where "ReLU" was selected as the activation function. The structure of the network (number of layers and neurons) was selected after cross-validation.

Theoretically, the length of the sequence of features that is provided as input to the neural network can be as large as the total number of trading steps in the optimisation horizon. In practice though, there are limitations with respect to the memory that is required to store a tensor of this size. As we can observe in Figure 5, each sample in the batch contains a vector of size 263 for each time-step. Assuming a certain batch size, there is a certain limit to the number of steps that can be stored in the memory. Therefore, for practical reasons and due to hardware limitations, we assume a history length $\bar{h}$ defined as $z_{i,t} = (a'_{i,t-\bar{h}-1}, r_{i,t-\bar{h}-1}, s'_{i,t-\bar{h}}, a'_{i,t-\bar{h}}, r_{i,t-\bar{h}}, ..., a'_{i,t-1}, r_{i,t-1}, s'_{i,t}) \in Z_i$. At each step t, the history length $\bar{h}$ takes the minimum value between the time-step t and $\bar{h}_{max}$, ($\bar{h} = min(t, \bar{h}_{max})$). Additionally, we provide the variable $\bar{s}_t = (a'_{i,t-1}, r_{i,t-1}, s'_{i,t})$, as a fixed size input for each step t of the LSTM. Consequently, the pseudo-state can be written as $z_{i,t} = (\bar{s}_{t-\bar{h}}, ..., \bar{s}_t)$.

4.8. Asynchronous Distributed Fitted Q iteration

The exploration requirements of the continuous state space, as defined previously introduce the necessity for collecting a large number of trajectories M. The total time required for gathering these trajectories heavily depends on the simulation time needed for one episode. In this particular setting developed, the simulation time can be quite long since, at each decision step, if the action selected is "Trade", an optimisation model is constructed and solved.

In this paper and in order to address this issue, we resort to an asynchronous architecture, similar to the one proposed in (Horgan et al., 2018), presented in Figure 6. The two processes, described in Section 4.6, namely generation of trajectories and computation of the Q-functions, run concurrently with no high-level synchronization.

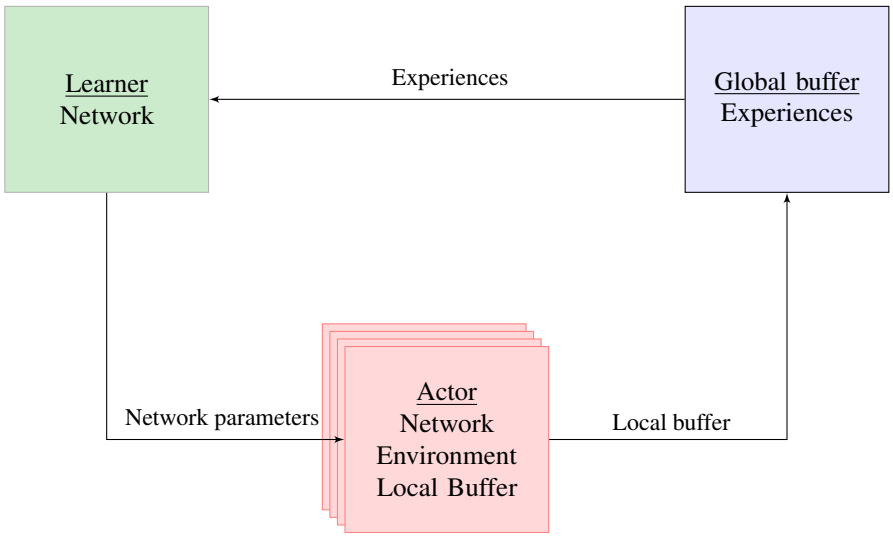


Figure 6. Schematic of the asynchronous distributed architecture. Each actor runs on a different thread and contains a copy of the environment, an individual ε-greedy policy based on the latest version of the network parameters and a local buffer. The actors generate trajectories that are stored in their local buffers. When the local buffer of each actor is filled, it is appended to the global buffer and the agent collects the latest network parameters from the learner. A single learner runs on a separate thread and is continuously training using experiences from the global buffer.

Multiple actors that run on different threads are used to generate trajectories. Each actor contains a copy of the environment, an individual ε-greedy policy based on the latest version of the Q functions and a local buffer. The actors use their ε-greedy policy to perform transitions in the environment. The transitions are stored in the local buffer. When the local buffer of each actor is filled, it is appended to the global buffer, the agent collects the latest Q-functions from the learner and continues the simulation. A single learner continuously updates the Q-functions using the simulated trajectories from a global buffer.

The benefits from asynchronous methods in Deep Reinforcement Learning (DRL) are elaborated in (Mnih et al., 2016). Each actor can use a different exploration policy (different initial ε value and decay) in order to enhance diversity in the collected samples which leads to a more stable learning process. Additionally, it is shown that the total computational time scales linearly with the number of threads considered. Another major advantage is that distributed techniques were shown to have a super-linear speedup for one-step methods that are not only related to computational gains. It is argued that, the positive effect of having multiple threads leads to a reduction of the bias in one-step methods (Mnih et al., 2016). In this way, these algorithms are shown to be much more data efficient than the original versions.

5. Case study

The proposed methodology is applied for the case of a PHES unit. First, the parameters and the exogenous information used for the optimisation of the CID market participation of a PHES operator are described. Second, the benchmark strategy used for comparison purposes and the process that was carried out for validation are presented. Finally, performance results of the obtained policy are presented and discussed.

5.1. Parameters specification

The proposed methodology is applied for a PHES unit participating in the German CID market with the following characteristics:

- $\overline{SoC_i} = 200$ MWh,
- $\underline{SoC_i} = 0$ MWh,
- $SoC_i^{init} = SoC_i^{term} = \left(\overline{SoC_i} - \underline{SoC_i}\right)/2$,
- $\overline{C_i} = \overline{G_i} = 200$ MW,
- $\underline{C_i} = \underline{G_i} = 0$ MW,
- $\eta = 100\%$.

The discrete trading horizon has been selected to be ten hours, i.e. $T = \{17:00, ..., 03:00\}$. The trading time interval is selected to be $\Delta t = 15$ min. Thus the trading process takes $K = 40$ steps until termination. Moreover, all 96 quarter-hourly products of the day, $X = \{Q_1, .., Q_{96}\}$, are considered. Consequently, the delivery timeline is $\bar{T} = \{00:00, ..., 23:45\}$, with $\tau_{init} = 00:00$ and $\tau_{term} = 24:00$

and the delivery time interval is $\Delta\tau = 15$ min. Each product can be traded until 30 minutes before the physical delivery of electricity begins (e.g. $t_{close}(Q_1) = 23:30$ etc.).

The number of days selected for training was $|L^{train}| = 252$ days. For the construction of the training/test sets, the days were sampled uniformly without replacement from a pool of 362 days. The number of simulated episodes for each training day was selected to be $E = 2000$ episodes for the artificial trajectories generation process, described in Section 4.6. During the trajectories generation process the high-level actions ("Trade" or "Idle") were chosen following an ε-greedy policy. As described in Section 4.8, each of the actor threads is provided with a different exploration parameter ε that is initialised with a random uniform sample in the range $[0.1, 0.5]$. The parameter ε is then annealed exponentially until a zero value is reached.

The pseudo-state $z_{i,t} = (s'_{i,0}, a'_{i,0}, r_{i,0}, ..., a'_{i,t-1}, r_{i,t-1}, s'_{i,t}) \in Z_i$ is composed of the entire history of observations and actions up to time-step t, as described in Section 4.5. For the sake of memory requirements, as explained in Section 4.7, we assume that the last ten trading steps contain sufficient information about the past. Thus, the pseudo-state is transformed in sequences of fixed length $\bar{h}_{max} = 10$,

5.2. Exogenous variable

The exogenous variable $w^{exog}_{i,t}$ represents any relevant information available to agent i about the system. In this case study, we assumed that the variable $w^{exog}_{i,t}$ contains:

- The 24 values of the Day-ahead price for the entire trading day
- The Imbalance price and the system Imbalance for the four quarters preceding each time-step t
- Time features: i) the hour of the day, ii) the month and iii) whether the traded day is a weekday or weekend

5.3. Benchmark strategy

The strategy selected for comparison purposes is the "rolling intrinsic" policy, denoted by π^{RI} (Lohndorf & Wozabal, 2015). According to this policy, the agent selects at each trading time-step t the action "Trade", as described in Section 4.4. This benchmark is selected since it represents the current state of the art used in the industry for the optimisation of PHES unit market participation.

5.4. Validation process

The performance of the policy obtained using the fitted Q algorithm, denoted by π^{FQ}, is evaluated on test set L^{test} that contains historical data from 110 days. These days are not used during the training process. This process of backtesting a strategy on historical data is widely used because it can provide a measure of how successful a strategy would be if it had been executed in the past. However, there is no guarantee that this performance can be expected in the future. This validation process heavily relies on Assumption (8) about the inability of the agent to influence the behaviour of the other players in the market. It can still provide an approximation on the results of the obtained policy before deploying it in real life. However, the only way to evaluate the exact viability of a strategy is to deploy it in real life.

We compare the performances of the policy obtained by the fitted Q iteration algorithm π^{FQ} and the "rolling intrinsic" policy π^{RI}. The comparison will always be based on the computation of the return of the policies on each day. For a given policy, the return over a day will be simply computed by running the policy on the day and summing up the rewards obtained.

The learning algorithm that we have designed for learning a policy has two sources of variance, namely those related to the generation of the new trajectories and those related to the learning of the Q-functions from the set of trajectories. To evaluate the performance of our learning algorithm we will perform several runs and average the performances of the policies learned. In the following, when we report the performance of a fitted Q iteration policy over one day, we will actually report the average performances of ten learned policies over this day.

We now describe the different indicators that will be used afterwards to assess the performances of our method. These indicators will be computed for both the training set and the test set, but are detailed hereafter when they are computed for the test set. It is straightforward to adapt the procedure for computing the indicators for the training set.

Let $V_d^{\pi^{FQ}}$ and $V_d^{\pi^{RI}}$ denote the total return of the fitted Q and the "rolling intrinsic" policy for day d, respectively. We gather the obtained returns of each policy for each day $d \in L^{test}$. We sort the returns in ascending order, and we obtain an ordered set containing a number of $|L^{test}|$ values for each policy. We provide descriptive statistics about the distribution of the returns of each policy $V_d^{\pi^{FQ}}$ and $V_d^{\pi^{RI}}$ on the test set L^{test}. In particular, we report the mean, the minimum and maximum values achieved for the set considered. Moreover, we provide the values obtained for each of the quartiles (25%, 50% and 75%) of the set.

Additionally, we compute the sum of returns over the entire set of days as follows:

$$V^{\pi^{FQ}} = \sum_{d \in L^{test}} V_d^{\pi^{FQ}}, \tag{52}$$

$$V^{\pi^{RI}} = \sum_{d \in L^{test}} V_d^{\pi^{RI}}. \tag{53}$$

An alternative performance indicator considered is the discrepancy of the returns coming from the fitted Q policy with respect to the risk-averse rolling intrinsic policy. We define the profitability ratio r_d for each day $d \in L^{test}$, that corresponds to the signed percentage difference between the two policies as follows:

$$r_d = \frac{V_d^{\pi^{FQ}} - V_d^{\pi^{RI}}}{V_d^{\pi^{RI}}} \cdot 100\%. \tag{54}$$

In a similar fashion, we sort the profitability ratios obtained for each day in the test set and we provide descriptive statistics about its distribution across the set. The mean, minimum and maximum values of the profitability ratio as well as the values of each quartile are reported. Finally, we compute the profitability ratio for the sum of returns over the entire set between the two policies, as:

$$r_{sum} = \frac{V^{\pi^{FQ}} - V^{\pi^{RI}}}{V^{\pi^{RI}}} \cdot 100\%. \tag{55}$$

5.5. Results

The performance indicators described previously are computed for both the training and the test set. The results obtained are summarised in Tables 3 and 4. Descriptive statistics about the distribution of the returns from both policies as well as the profitability ratio are presented for each dataset.

It can be observed that on average π^{FQ} yields better returns than π^{RI} both on the training and the test set. More specifically, on the training set, the obtained policy performs, on average 2.56% better than the "rolling intrinsic" policy. For the top 25% of the training days the profitability ratio is higher than 3.69% and in some cases it even exceeds 10%. Overall, the total profits coming from the fitted Q policy add up to €2,167,624, yielding a difference of €57,423.3 (2.7%) more than the profits from the "rolling intrinsic" for the set of 252 days considered.

The fitted Q policy yields on average a 1.5% greater profit on the test set with respect to the returns of the "rolling intrinsic" policy. It is important to highlight that for 50% of the test set, the profits from the fitted Q policy are higher than 1% in comparison to the "rolling intrinsic". The difference between the total profits resulting from the two policies over the set of 110 days considered amounts to €15,681 (1.7%).

The distribution of training and test set samples according to the obtained profitability ratio is presented in Figure 7. It can be observed that most samples are spread in the interval between $0-5\%$ and that the distribution has a positive skew.

Table 3. Descriptive statistics of the returns obtained on the days of the training set for policies π^{FQ} and π^{RI}. The last column also provides the corresponding profitability ratios.

	π^{FQ} returns (€)	π^{RI} returns (€)	r (%)
mean	8670	8440	2.56
min	2586	2524	−0.42
25%	5571	5537	0.33
50%	7496	7385	1.45
75%	11081	10632	3.69
max	54967	54285	23.12
sum	2167624	2110200	2.7

Table 4. Descriptive statistics of the returns obtained on the days of the test set for policies π^{FQ} and π^{RI}. The last column also provides the corresponding profitability ratios.

	π^{FQ} returns (€)	π^{RI} returns (€)	r (%)
mean	8583	8439	1.50
min	2391	2366	−0.73
25%	5600	5527	0.23
50%	7661	7622	0.87
75%	10823	10721	2.22
max	37902	36490	6.56
sum	935552	919871	1.7

It is important to note that, for the vast majority of the samples, the profitability ratio is nonnegative. This result allows us to characterize the new policy as risk-free in the sense that in the worst case, the fitted Q policy performs as well as the "rolling intrinsic" policy. From the standpoint of practical implementation this result allows us to construct a wrapper around the current industrial standard practices and expect an average improved performance of 2% almost without any risk. However, as discussed earlier, the backtesting of a strategy in historical data may differ from the outcomes in real deployment for various reasons.

6. Discussion

In this section, we provide some remarks related to the practical challenges encountered and the validity of the assumptions considered throughout this paper.

6.1. Behaviour of the rest of the agents

In this paper, we assumed (Assumption 1) that the rest of the agents $-i$ post orders in the market based on their needs and some historical information of the state of the order book. In reality, the available information that the other agents possess is not accessible by agent i. This fact gives rise to issues related to the validity of the assumption that the process is Markovian.

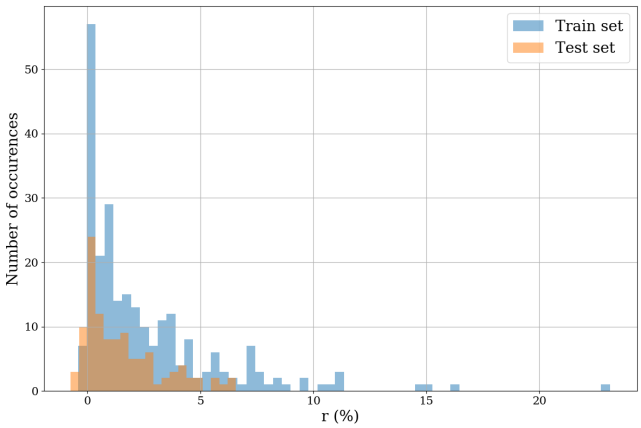


Figure 7. Profitability ratio.

We further assumed (8) in Section 4.6 that the behaviour of agent i does not influence the strategy of the other agents $-i$. Based on this assumption the training and the validation process were performed using historical data. However, the strategy of each of the market participants is highly dependent on the actions of the rest participants, especially in a market with limited liquidity such as the CID market.

These assumptions, although slightly unrealistic and optimistic, provide us with a meaningful testing protocol for a trading strategy. The actual profitability of a strategy can be obtained by deploying the strategy in real-time. However, it is important to show that the strategy is able to obtain substantial profits in back-testing first.

6.2. Partial observability of the process

In Section 3, the decision-making problem studied in this paper was framed as an MDP after considering certain assumptions. Theoretically, this formulation is very convenient, but does not hold in practice. Especially considering Assumption 7, where the reduced pseudo-state is assumed to contain all the relevant information required.

Indeed, the trading agents do not have access to all the information required. For instance, a real agent does not know how many other agents are active in the market. They do not know the strategy of each agent either. There is also a lot of information gathered by w^{exog} which is not available for the agent. Finally, the fact that the state space was reduced results in an inevitable loss of information.

Therefore, it would be more accurate to consider a Partially Observable Markov Decision Process (POMDP) instead. In a POMDP, the real state is hidden and the agent only has access to observations. For an RL algorithm to properly work with a POMDP, the observations have to be representative of the real hidden states.

6.3. Exploration

There are two main issues related to the state space exploration that result in the somewhat limited performance of the obtained policy. First, in the described setting, the way in which we generate the artificial trajectories is very important for the success of the method. The generated states must be "representative" in the sense that the areas around these states are visited often under a near optimal policy (Bertsekas, 2005). In particular, the frequency of appearance of these areas of states in the training process should be proportional to the probability of occurrence under the optimal policy. However, in practice, we are not in a position to know which areas are visited by the optimal policy. In that respect, the asynchronous distributed algorithm used in this paper was found to successfully address the issue of state exploration.

Second, the assumptions (Assumptions 3, 4, 5) related to the operation of the storage device according to the "default" strategy without any imbalances allowed, as well as the participation of the agent as an aggressor, are restrictive with respect to the set of all admissible policies. Additionally, the adoption of the reduced discrete action space described in Section 4.4 introduces further restrictions on the set of available actions. Although having a small and discrete space is convenient for the optimisation process, it leads to limited state exploration. For instance, the evolution of the state of charge of the storage device is always given as the output of the optimisation model based on the order book data. Thus, in this configuration, it is not possible to explore all areas of the state space (storage levels) but only a certain area driven by the historical order book data. However, evaluating the policy on a different dataset might lead to areas of the state space (e.g. storage level) that are never visited during training, leading to poor performance. Potential mitigations of this issue involve diverse data augmentation techniques and/or different representation of the action space.

7. Conclusions and future work

In this paper, a novel RL framework for the participation of a storage device operator in the CID market is proposed. The energy exchanges between market participants occur through a centralized order book. A series of assumptions related to the behaviour of the market agents and the operation of the storage device are considered. Based on these assumptions, the sequential decision-making problem is cast as an MDP. The high dimensionality of both the action and the state spaces increase the computational complexity of finding a policy. Thus, we motivate the use of discrete high-level actions that map into the original action space. We further propose a more compact state representation. The resulting decision process is solved using fitted Q iteration,

a batch mode reinforcement learning algorithm. The results illustrate that the obtained policy is a low-risk policy that is able to outperform on average the state of the art for the industry benchmark strategy ("rolling intrinsic") by 1.5% on the test set. The proposed method can serve as a wrapper around the current industrial practices that provides decision support to energy trading activities with low risk.

The main limitations of the developed strategy originate from: i) the insufficient amount of relevant information contained in the state variable, either because the state reduction proposed leads to a loss of information or due to the unavailability of information and ii) the limited state space exploration as a result of the proposed high-level actions in combination with the use of historical data. To this end and as future work, a more detailed and accurate representation of the state should be devised. This can be accomplished by increasing the amount of information considered, such as RES forecasts, and by improving the order book representation. We propose the use of continuous high-level actions in an effort to gain state exploration without leading to very complex and high-dimensional action space.

References

Aïd, R., Gruet, P., and Pham, H. An optimal trading problem in intraday electricity markets. *Mathematics and Financial Economics*, 10(1):49–85, 2016.

Baillo, A., Ventosa, M., Rivier, M., and Ramos, A. Optimal offering strategies for generation companies operating in electricity spot markets. *IEEE Transactions on Power Systems*, 19(2):745–753, May 2004. ISSN 0885-8950. doi: 10.1109/TPWRS.2003.821429.

Balardy, C. German continuous intraday market: Orders book's behavior over the trading session. In *Meeting the Energy Demands of Emerging Economies, 40th IAEE International Conference, June 18-21, 2017*. International Association for Energy Economics, 2017a.

Balardy, C. An analysis of the bid-ask spread in the german power continuous market. In *Heading Towards Sustainable Energy Systems: Evolution or Revolution?, 15th IAEE European Conference, Sept 3-6, 2017*. International Association for Energy Economics, 2017b.

Bertrand, G. and Papavasiliou, A. Adaptive trading in continuous intraday electricity markets for a storage unit. *IEEE Transactions on Power Systems*, pp. 1–1, 2019. ISSN 1558-0679. doi: 10.1109/TPWRS.2019.2957246.

Bertsekas, D. P. *Dynamic programming and optimal control*, volume 1. Athena scientific Belmont, MA, 2005.

Boomsma, T. K., Juul, N., and Fleten, S.-E. Bidding in sequential electricity markets: The Nordic case. *European Journal of Operational Research*, 238(3):797 – 809, 2014. ISSN 0377-2217. doi: http://dx.doi.org/10.1016/j.ejor.2014.04.027. URL http://www.sciencedirect.com/science/article/pii/S0377221714003695.

Borggrefe, F. and Neuhoff, K. Balancing and intraday market design: Options for wind integration. DIW Discussion Papers 1162, Berlin, 2011. URL http://hdl.handle.net/10419/61319.

Braun, S. and Hoffmann, R. Intraday Optimization of Pumped Hydro Power Plants in the German Electricity Market. In *Energy Procedia*, volume 87, pp. 45–52, 2016. doi: 10.1016/j.egypro.2015.12.356.

Busoniu, L., Babuska, R., De Schutter, B., and Ernst, D. *Reinforcement learning and dynamic programming using function approximators*. CRC press, 2017.

EPEXSPOT. Market data intraday continuous, 2017. URL http://www.epexspot.com/en/market-data/intradaycontinuous.

EPEXSPOT. EPEXSPOT Operational rules, 2019. URL https://www.epexspot.com/document/40170/EPEX%20SPOT%20Market%20Rules.

Ernst, D., Geurts, P., and Wehenkel, L. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6(Apr):503–556, 2005.

Fleten, S. E. and Kristoffersen, T. K. Stochastic programming for optimizing bidding strategies of a Nordic hydropower producer. *European Journal of Operational Research*, 181(2):916 – 928, 2007. ISSN 0377-2217. doi: http://dx.doi.org/10.1016/j.ejor.2006.08.023.

Garnier, E. and Madlener, R. Balancing forecast errors in continuous-trade intraday markets. *Energy Systems*, 6(3):361–388, 2015.

Gönsch, J. and Hassler, M. Sell or store? An ADP approach to marketing renewable energy. *OR Spectrum*, 38(3):633–660, 2016. ISSN 14366304. doi: 10.1007/s00291-016-0439-x.

Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

Hagemann, S. Price determinants in the German intraday market for electricity: an empirical analysis. *Journal of Energy Markets*, 2015.

Hassler, M. Heuristic decision rules for short-term trading of renewable energy with co-located energy storage. *Computers and Operations Research*, 83, 2017. ISSN 03050548. doi: 10.1016/j.cor.2016.12.027.

Henriot, A. Market design with centralized wind power management: Handling low-predictability in intraday markets. *Energy Journal*, 35(1):99–117, 2014. ISSN 01956574. doi: 10.5547/01956574.35.1.6.

Horgan, D., Quan, J., Budden, D., Barth-Maron, G., Hessel, M., Van Hasselt, H., and Silver, D. Distributed prioritized experience replay. *arXiv preprint arXiv:1803.00933*, 2018.

Jiang, D. R. and Powell, W. B. Optimal hour-ahead bidding in the real-time electricity market with battery storage using Approximate Dynamic Programming. pp. 1–28, 2014. ISSN 1091-9856. doi: 10.1287/ijoc.2015.0640. URL http://arxiv.org/abs/1402.3575.

Karanfil, F. and Li, Y. The role of continuous intraday electricity markets: The integration of large-share wind power generation in Denmark. *Energy Journal*, 38(2): 107–130, 2017. ISSN 01956574. doi: 10.5547/01956574.38.2.fkar.

Le, H. L., Ilea, V., and Bovo, C. Integrated European intra-day electricity market: Rules, modeling and analysis. *Applied Energy*, 238(June 2018):258–273, 2019. ISSN 03062619. doi: 10.1016/j.apenergy.2018.12.073. URL https://doi.org/10.1016/j.apenergy.2018.12.073.

Lohndorf, N. and Wozabal, D. Optimal gas storage valuation and futures trading under a high-dimensional price process. Technical report, Technical report, 2015.

Meeus, B. L. and Schittekatte, T. The EU Electricity Network Codes: Course text for the Florence School of Regulation online course. (October), 2017.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937, 2016.

Neuhoff, K., Ritter, N., Salah-Abou-El-Enien, A., and Vassilopoulos, P. Intraday markets for power: discretizing the continuous trading? 2016.

PandŽić, H., Morales, J. M., Conejo, A. J., and Kuzle, I. Offering model for a virtual power plant based on stochastic programming. *Applied Energy*, 105:282–292, 2013. ISSN 03062619. doi: 10.1016/j.apenergy.2012.12.077.

Pérez Arriaga, I. and Knittel et al, C. *Utility of the future. An MIT Energy Initiative response*. 2016. ISBN 9780692808245. URL energy.mit.edu/uof.

Plazas, M. A., Conejo, A. J., and Prieto, F. J. Multimarket optimal bidding for a power producer. *IEEE Transactions on Power Systems*, 20(4):2041–2050, Nov 2005. ISSN 0885-8950. doi: 10.1109/TPWRS.2005.856987.

Scharff, R. and Amelin, M. Trading behaviour on the continuous intraday market Elbas. *Energy Policy*, 88:544–557, 2016. ISSN 03014215. doi: 10.1016/j.enpol.2015.10.045.

Spot, N. Xbid cross-border intra day market project, 2018. URL https://www.nordpoolspot.com/globalassets/download-center/xbid/xbid-qa_final.pdf.

The European Commission. 2030 energy strategy, 2017. URL https://ec.europa.eu/energy/en/topics/energy-strategy-and-energy-union/2030-energy-strategy.

Watkins, C. J. and Dayan, P. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

A. Nomenclature

Acronyms

ADP	Approximate Dynamic Programming.
CID	Continuous Intraday.
DRL	Deep Reinforcement Learning.
FCFS	First Come First Served.
MDP	Markov Decision Process.
OB	Order Book.
PHES	Pumped Hydro Energy Storage.
RES	Renewable Energy Sources.

Sets and indexes

Name	Description
i	Index of an agent.
$-i$	Index of all the agents except agent i.
j	Index of an order.
m	Index of a sample of quadruples.
d	Index of a day in a set.
t	Trading time-step.
τ	Discrete time-step of delivery.
A	Joint action space for all the agents.
A_i	Action space of agent i.
A_{-i}	Action space of the rest of the agents $-i$.
A_i^{red}	Reduced action space of agent i.
A_i'	Set of high-level actions for agent i.
$\bar{A}_i$	Set of all factors for the partial/full acceptance of orders by agent i.

E	Set of conditions that can apply to an order.
F	Set of all sampled trajectories.
F'	Set of sampled one-step transitions.
F'_t	Set of sampled one-step transitions for time t.
H_i	Set of all histories for agent i.
I	Set of agents.
L^{train}	Set of trading days used to train the agent.
L^{test}	Set of trading days used to evaluate the agent.
N_t	Set of all available order unique indexes at time t.
N'_t	Set of all the unique indexes of new orders posted at time t.
N_τ	Set of all the unique indexes of orders for delivery at τ.
O_t	Set of all available orders in the order book at time t.
S^{OB}	Set of all available orders in the order book.
S'^{OB}	Low dimensional set of all available orders in the order book.
S_i	State space of agent i.
T	Trading horizon, i.e. time interval between first possible trade and last possible trade.
$T(x)$	Discretization of the trading timeline for product x.
$\bar{T}$	Discretization of the delivery timeline.
$\bar{T}(t)$	Discretization of the delivery timeline at trading step t.
T^{Imb}	Discretization of the imbalance settlement timeline.
X	Set of all available products.
X_t	Set of all available products at time t.
Z_i	Set of pseudo-states for agent i.
Π	Set of all admissible policies.

Parameters

Name	Description
$\overline{C_i}$	Maximum consumption level for the asset of agent i.
$\underline{C_i}$	Minimum consumption level for the asset of agent i.
E	Number of episodes.
e	Conditions applying on an order other than volume and price.
ep	Number of simulations between two successive Q function updates.
$decay$	Parameter for the annealing of ε.
$\overline{G_i}$	Maximum production level for the asset of agent i.
$\underline{G_i}$	Minimum production level for the asset of agent i.
$\bar{h}$	Sequence length of past information.
$\bar{h}_{max}$	Maximum sequence length of past information.
$I(\tau)$	Imbalance price for delivery period $\delta(x)$.
K	Number of steps in the trading period.
M	Number of samples of quadruples.
n	Number of agents.
o_t	Market order.
p	Price of an order.
$p_{\max}$	Maximum price of an order.
$p_{\min}$	Minimum price of an order.
$\overline{SoC_i}$	Maximum state of charge of storage device.
$\underline{SoC_i}$	Minimum state of charge of storage device.
$\overline{SoC_i^{init}}$	State of charge of storage device at the beginning of the delivery timeline.
SoC_i^{term}	State of charge of storage device at the end of the delivery timeline.
$t_{close}(x)$	End of trading period for product x.
$t_{delivery}(x)$	Start of delivery of product x.
$t_{open}(x)$	Start of trading period for product x.
$t_{settle}(x)$	Time of settlement for product x.
v	Volume of an order.
x	Market product.
y	Side of an order ("Sell" or "Buy").
y_t^m	Target computed for sample m at time t.
$\delta(x)$	Time interval covered by product x (delivery).
Δt	Time interval between trading time-steps.
$\Delta\tau$	Time interval between delivery time-steps.
ε	Parameter for the ε-greedy policy.
η	Charging/discharging efficiency of storage device.
θ_t	Parameters vector of function approximation at time t.
$\lambda(x)$	Duration of time-interval $\delta(x)$.
ζ	A single trajectory.
ζ_m	A single indexed trajectory.
τ_{init}	Initial time-step of the delivery timeline.
τ_{term}	Terminal time-step of the delivery timeline.

Variables

Name	Description
a_t	Joint action from all the agents at time t.
$a_{i,t}$	Action of posting orders by agent i at time t.
$a_{-i,t}$	Action of posting orders by the rest of the agents $-i$ at time t.
$a'_{i,t}$	High-level action by agent i at time t.
$a^j_{i,t}$	Acceptance (partial/full) factor for order j by agent i at time t.
$\bar{a}_{i,t}$	Factors for the partial/full acceptance of all orders by agent i at time t.
$C_{i,t}(\tau)$	Consumption level at delivery time-step τ computed at time t.
$c_{i,t}(t')$	Consumption level during the delivery interval.
$e_{i,t}$	Random disturbance for agent i at time t.

$G_{i,t}(\tau)$	Generation level at delivery time-step τ computed at t.
$g_{i,t}(t')$	Generation level during the delivery interval.
$h_{i,t}$	History vector of agent i at time t.
$k_{i,t}(\tau)$	Binary variable that enforces either charging or discharging of the storage device.
$P^{mar}_{i,t}(x)$	Net contracted power of agent i for product x (delivery time-step τ) at time t.
$P^{res}_{i,t}(\tau)$	Residual production of agent i delivery time-step τ (for product x) at time t.
$P^{res}_{i}(\tau)$	Final residual production of agent i for product
$r_{i,t}$	Instantaneous reward of agent i at time t.
r_d	Profitability ratio at day d.
r_{sum}	Profitability ratio for the sum of returns over set.
$s_{i,t}$	State of agent i at time t.
$SoC_{i,t}(\tau)$	State of charge of device at delivery time-step τ computed at t.
s^{OB}_t	State of the order book at time t.
s'^{OB}_t	Low dimensional state of the order book at time t.
$s^{private}_{i,t}$	Private information of agent i at time t.
$\bar{s}_t$	Triplet of fixed size, part of pseudo-state $z_{i,t}$ that serves as an input at LSTM at time t.
$u_{i,t}$	Aggregate (trading and asset) control action of the asset trading agent i at time t.
$v^{con}_{i,t}(x)$	Volume of product x contracted by agent i at time t.
$w^{exog}_{i,t}$	Exogenous information of agent i at time t.
$z_{i,t}$	Pseudo-state for agent i at time t.
$\Delta_{i,t}(\tau)$	Imbalance for delivery time τ for agent i computed at time t.
$\Delta_i(\tau)$	Final imbalance for delivery time τ for agent i.
$\Delta G_{i,t}$	Change in the production level for the asset of agent i at time t.
$\Delta C_{i,t}$:	Change in the consumption level for the asset of agent i at time t.

Functions

Name	Description
$clear(\cdot)$	Market clearing function.
$b(\cdot)$	Univariate stochastic model for exogenous information.
$e(\cdot)$	Encoder that maps from the original state space H_i to pseudo-state space Z_i.
$f(\cdot)$	Order book transition function.
$G^{\zeta}(\cdot)$	Revenue collected over a trajectory.
$g(\cdot)$	System dynamics of the MDP.
$k(\cdot)$	System dynamics of asset trading process.
$l(\cdot)$	Reduced action space construction function.
$P_{a_{-i,t}(\cdot)}$	Probability distribution function for the actions of the rest of the agents $-i$.
$P_{e_t}(\cdot)$	Random disturbance probability distribution function.
$P(\cdot)$	Transition probabilities of the MDP.
$P_{FQ}(\cdot)$	The stochastic process (algorithm) of fitted Q iteration.
$P_{\theta_{t,0}}(\cdot)$	Distribution of the initial parameters $\theta_{t,0}$.
$p(\cdot)$	Mapping from high-level actions A'_i to the reduced action space A^{red}_i.
$Q_t(\cdot,\cdot)$	State-action value function at time t.
$\hat{Q}(\cdot,\cdot)$	Sequence of Q-function approximations.
$R(\cdot)$	Reward function.
$u(\cdot)$	Signing convention for the volume wrt. the side ('Buy" or 'Sell") of each order.
$V^{\pi_i}(\cdot)$	Total expected reward function for policy π_i.
$V^{\pi^{FQ}}_d(\cdot)$	Return of the fitted Q policy π^{FQ}_i for day d.
$V^{\pi^{RI}}_d(\cdot)$	Return of the "rolling" intrinsic policy π^{RI}_i for day d.
$\mu_t(\cdot)$	Policy function at time t.
$\pi_i(\cdot)$	Policy followed by agent i.
$\rho(\cdot)$	Trading revenue function.

3 Learning optimal environments using projected stochastic gradient ascent

Learning optimal environments using projected stochastic gradient ascent

Adrien Bolland
Montefiore Institute
University of Liège
Liège, Belgium
adrien.bolland@student.uliege.be

Ioannis Boukas
Montefiore Institute
University of Liège
Liège, Belgium
ioannis.boukas@uliege.be

François Cornet
Montefiore Institute
University of Liège
Liège, Belgium
f.cornet@student.uliege.be

Mathias Berger
Montefiore Institute
University of Liège
Liège, Belgium
mathias.berger@uliege.be

Damien Ernst
Montefiore Institute
University of Liège
Liège, Belgium
dernst@uliege.be

Abstract

In this work, we generalize the direct policy search algorithms to an algorithm we call Direct Environment Search with (projected stochastic) Gradient Ascent (DESGA). The latter can be used to jointly learn a reinforcement learning (RL) environment and a policy with maximal expected return over a joint hypothesis space of environments and policies. We illustrate the performance of DESGA on two benchmarks. First, we consider a parametrized space of Mass-Spring-Damper (MSD) environments. Then, we use our algorithm for optimizing the size of the components and the operation of a small-scale and autonomous energy system, i.e. a solar off-grid microgrid, composed of photovoltaic panels, batteries, etc. The results highlight the excellent performances of the DESGA algorithm.

1 Introduction

In reinforcement learning (RL), an active decision-making agent attempts to learn a policy in order to maximize its value function, through interaction with its environment. During this interaction, the agent collects experience, that is used to improve its performance over time. The goal of the agent is defined by the reward signal collected after each interaction with the environment.

In this paper, we extend the standard RL framework by considering that, in addition to the policy, the environment (transition dynamics and reward signal) is parametrized. The objective of our approach is to jointly optimize the environment and the policy parameters in order to maximize the total expected rewards received. This type of problem is encountered in many fields where one has to design a system which has to be controlled afterwards. By way of example, we can mention the combined design and control of a robotic arm for achieving a specific goal [Castejón et al., 2010, Ajwad et al., 2018] or the sizing and the operation of a microgrid to minimize electricity costs [François-Lavet et al., 2016]. For

Preprint. Under review.

the optimization, we propose an algorithm that generalizes the direct policy search algorithms to an algorithm we call Direct Environment Search with (projected stochastic) Gradient Ascent (DESGA). This algorithm optimizes the different parameters by performing stochastic gradient ascent updates that are projected at every iteration on the feasible set of parameters.

Direct policy search techniques, extensively used in RL, parametrize the policy, and navigate in the space of candidate policies towards an optimal one by processing the information contained in trajectories that are generated throughout the optimization process. Typically, two main classes of direct policy search techniques can be distinguished, namely gradient-free and gradient-based methods. The first class uses derivative-free optimisation techniques, e.g. the covariance matrix adaptation (CMA) [Hansen and Ostermeier, 1996] and the cross-entropy method (CEM) [Szita and Lörincz, 2006, Buşoniu et al., 2011]. The latter class of methods moves from one point to the next, in the space of candidate policies, through the reconstruction of a gradient of the objective from information contained in trajectories.

Derivative-free methods are known to scale unfavourably with the number of policy parameters and do not perform well on large-scale problems [Schulman et al., 2015]. Gradient descent (or ascent) methods have been very successful at learning function approximators for supervised learning tasks with a large number of parameters [Kingma and Ba, 2014]. Gradient-based direct policy search methods extend their success to reinforcement learning and allow for efficient training of complex and powerful policies. The first gradient-based direct policy search technique, called the REINFORCE algorithm [Williams, 1992], has served as a basis for developing our DESGA algorithm.

The rest of the paper is organized as follows. In Section 2, we present the theoretical background and the problem statement of optimizing over the joint environment and policy parameter space. In Section 3, the proposed methodology as well as the algorithmic implementation for direct environment search with gradient ascent (DESGA) are described. The experimental protocol for the evaluation of the proposed algorithm and the results are demonstrated in Section 4. Finally, the conclusions and certain considerations for further work are discussed in Section 5.

2 Theoretical background and problem statement

In this section, we provide a generic formulation for the optimal control problem of a discrete-time dynamical system with a finite-time optimization horizon. Then, we introduce a parametrization of both the dynamical system and the policy spaces. Subsequently, we formulate the problem of jointly optimizing the vector of parameters of the dynamical system and the policy with the goal to maximize the total expected rewards.

2.1 Discrete-time dynamical systems

Let us consider a discrete-time and time-invariant dynamical system defined as follows [Bertsekas, 2005]. Let $T \in \mathbb{N}$ be the optimization horizon referring to the number of decisions to be taken in the control process. The system is defined by a state space $\mathcal{S}$, an action space $\mathcal{A}$, a disturbance space Ξ, a transition function $f : \mathcal{S} \times \mathcal{A} \times \Xi \to \mathcal{S}$, a bounded reward function $\rho : \mathcal{S} \times \mathcal{A} \times \Xi \to R \subset \mathbb{R}$ and a conditional probability distribution P_ξ giving the probability $P(\xi_t|s_t, a_t)$ of drawing a disturbance $\xi_t \in \Xi$ when taking an action $a_t \in \mathcal{A}$ while being in a state $s_t \in S$. A probability measure P_0 yields the probability $P_0(s_0)$ of each state $s_0 \in \mathcal{S}$ to be the initial state. At time $t \in \{0, 1, \ldots, T-1\}$, the system moves from state $s_t \in \mathcal{S}$ to state $s_{t+1} \in \mathcal{S}$ under the effect of an action $a_t \in \mathcal{A}$ and a random disturbance $\xi_t \in \Xi$, drawn with probability $P_\xi(\xi_t|s_t, a_t)$, according the transition function f:

$$s_{t+1} = f(s_t, a_t, \xi_t) \; . \tag{1}$$

After each transition, a reward signal r_t is collected from the reward function according to $r_t = \rho(s_t, a_t, \xi_t)$ with $|r_t| \leq r_{max}$. The different elements of this optimal control problem are gathered in a tuple $(\mathcal{S}, \mathcal{A}, \Xi, P_0, f, \rho, P_\xi, T)$ referred to as the environment.[1]

We define a closed-loop policy $\pi \in \Pi$ as a function associating a probability distribution with support $\mathcal{A}$ to current state s_t of the system at a decision stage $t = 0, \ldots, T-1$. Applying the policy to the

[1]Let us note that from the environment $(\mathcal{S}, \mathcal{A}, \Xi, P_0, f, \rho, P_\xi, T)$, we can define an equivalent Markov Decision Process (MDP) with horizon T, state space $\mathcal{S}$, action space $\mathcal{A}$, initial probability distribution P_0,

dynamical system consists in sampling an action a_t with probability $\pi(a_t|s_t, t)$ at each time t. A trajectory $\tau = (s_0, a_0, \xi_0, a_1, \xi_1, \dots a_{T-1}, \xi_{T-1})$ contains the information collected from executing policy π over the horizon T. The cumulative reward $R(\tau)$ over trajectory τ can be computed as:

$$R(\tau) = \sum_{t=0}^{T-1} r_t \ , \tag{4}$$

where $r_t = \rho(s_t, a_t, \xi_t)$ and $s_{t+1} = f(s_t, a_t, \xi_t)$. The expected cumulative reward associated to a policy π, and to an initial state $s_0 \in \mathcal{S}$, is called the return of the policy and is given by:

$$V^{\pi}(s_0) = \mathop{\mathbb{E}}_{a_t \sim \pi(\cdot|s_t,t), \xi_t \sim P_\xi(\cdot|s_t,a_t)} \{\sum_{t=0}^{T-1} r_t\} \ . \tag{5}$$

An optimal policy $\pi^* \in \Pi$ is a policy, that maximizes the expected cumulative reward collected for every initial state and is defined as:

$$\pi^* \in \mathop{\text{argmax}}_{\pi \in \Pi} V^{\pi}(s) \quad , \forall s \in \mathcal{S} \ . \tag{6}$$

2.2 Problem statement: optimizing over a set of environments

We consider the environment $(\mathcal{S}, \mathcal{A}, \Xi, P_0, f_\psi, \rho_\psi, P_\xi, T)$, as defined in Section 2.1, with continuous state space $\mathcal{S} \subset \mathbb{R}^{d_\mathcal{S}}$, action space $\mathcal{A} \subset \mathbb{R}^{d_\mathcal{A}}$, disturbance space $\Xi \subset \mathbb{R}^{d_\Xi}$, distribution P_0 over the initial states and horizon T; where $d_\mathcal{S}, d_\mathcal{A}, d_\Xi \in \mathbb{N}$. The state, action and disturbance spaces are assumed to be compact. The transition and reward functions are two parametric functions f_ψ and ρ_ψ, parametrized by the vector ψ defined over the compact $\Psi \subset \mathbb{R}^{d_\Psi}$, with $d_\Psi \in \mathbb{N}$. Both functions are assumed continuously differentiable with respect to their parameters and to the state space for every action in $\mathcal{A}$ and every disturbance in Ξ. Additionally, we consider the parametric function π_θ to be a policy parametrized by the real vector θ in the compact $\Theta \subset \mathbb{R}^{d_\Theta}$, with $d_\Theta \in \mathbb{N}$, and continuously differentiable with respect to its parameters Θ and to its domain $\mathcal{S}$ for every action in $\mathcal{A}$ and for every time t. We want to identify a pair of parameter vectors (ψ, θ) such that the policy π_θ maximizes the expected return, on expectation over the initial states, in the environment $(\mathcal{S}, \mathcal{A}, \Xi, P_0, f_\psi, \rho_\psi, P_\xi, T)$. We thus want to solve the following optimization problem:

$$\psi^*, \theta^* \in \mathop{\text{argmax}}_{\psi \in \Psi, \theta \in \Theta} V(\psi, \theta) \tag{7}$$

$$V(\psi, \theta) = \mathop{\mathbb{E}}_{\substack{s_0 \sim P_0(\cdot) \\ a_t \sim \pi_\theta(\cdot|s_t,t) \\ \xi_t \sim P_\xi(\cdot|s_t,a_t)}} \{\sum_{t=0}^{T-1} r_t\} \tag{8}$$

$$s_{t+1} = f_\psi(s_t, a_t, \xi_t) \tag{9}$$

$$r_t = \rho_\psi(s_t, a_t, \xi_t) \ . \tag{10}$$

3 Direct environment search with gradient ascent

In this section, we address the problem defined in Section 2.2. First, we show in Section 3.1 that the expected cumulative reward is differentiable with respect to the parameters of the system and the policy if the different parametric functions and the disturbance probability function are continuously differentiable. In such a context, we derive an analytical expression of the gradient. The results are also extended for discrete action and disturbance spaces. We also derive the expression of an unbiased estimator of the gradient from the differentiation of a loss function built from Monte-Carlo simulations. In Section 3.2, we present our Direct Environment Search with (projected stochastic) Gradient Ascent (DESGA) algorithm that uses a projected stochastic gradient ascent for optimizing both the parameters of the environment and the policy.

---

reward probability distribution r and transition probability distribution p such that:

$$r(r_t|s_t, a_t) = \mathop{\mathbb{E}}_{\xi_t \sim P_\xi(\cdot|s_t,a_t)} \{\delta_{\rho(s_t,a_t,\xi_t)}(r_t)\} \quad , \forall s_t, \in \mathcal{S}, a_t \in \mathcal{A}, r_t \in R \tag{2}$$

$$p(s_{t+1}|s_t, a_t) = \mathop{\mathbb{E}}_{\xi_t \sim P_\xi(\cdot|s_t,a_t)} \{\delta_{f(s_t,a_t,\xi_t)}(s_{t+1})\} \quad , \forall s_t, s_{t+1} \in \mathcal{S}, a_t \in \mathcal{A} \ , \tag{3}$$

where $\delta_y(x)$ is a function returning one if and only if x equals y and zero otherwise.

3.1 Gradient for learning optimal environments

In Theorem 1, we first prove the differentiability of the expected cumulative reward with respect to the policy and the environment parameters, assuming the functions composing the environment and the policy are continuously differentiable. We then extend these results in a straightforward way to the case where $\mathcal{A}$ and/or Ξ are discrete in Corollary 1. Corollaries 2 and 3 finally give the expressions of the gradients.

Theorem 1. Let $(\mathcal{S}, \mathcal{A}, \Xi, P_0, f_\psi, \rho_\psi, P_\xi, T)$ and π_θ be an environment and a policy as defined in Section 2.2. Additionally, let the functions f_ψ, ρ_ψ and P_ξ be continuously differentiable over their domain of definition. Let $V(\psi, \theta)$ be the expected cumulative reward of policy π_θ, averaged over the initial states, for all $(\psi, \theta) \in \Psi \times \Theta$, as defined in Eqn. (8).

Then, the function V exists, is bounded, and is continuously differentiable in the interior of $\Psi \times \Theta$.

Corollary 1. The function V, as defined in Theorem 1, exists, is bounded, and is continuously differentiable in the interior of $\Psi \times \Theta$ if $\mathcal{A}$ and/or Ξ are discrete.

Corollary 2. The gradient of the function V defined in Eqn. (8) with respect to the parameter vector ψ is such that:

$$\nabla_\psi V(\psi, \theta) = \mathop{\mathbb{E}}_{\substack{s_0 \sim P_0(\cdot) \\ a_t \sim \pi_\theta(\cdot|s,t) \\ \xi_t \sim P_\xi(\cdot|s_t, a_t)}} \Big\{ \Big(\sum_{t=0}^{T-1} \big(\nabla_s \log \pi_\theta(a_t|s,t)|_{s=s_t} + \nabla_s \log P_\xi(\xi_t|s,a_t)|_{s=s_t} \big) \cdot \nabla_\psi s_t \Big) \times \Big(\sum_{t=0}^{T-1} r_t \Big) + \Big(\sum_{t=0}^{T-1} \nabla_\psi \rho_\psi(s, a_t, \xi_t)|_{s=s_t} + \nabla_s \rho_\psi(s, a_t, \xi_t)|_{s=s_t} \cdot \nabla_\psi s_t \Big\}, \quad (11)$$

where:

$$\nabla_\psi s_t = (\nabla_s f_\psi)(s, a_{t-1}, \xi_{t-1})|_{s=s_{t-1}} \cdot \nabla_\psi s_{t-1} + (\nabla_\psi f_\psi)(s, a_{t-1}, \xi_{t-1})|_{s=s_{t-1}} \ , \quad (12)$$

with $\nabla_\psi s_0 = 0$.

Corollary 3. The gradient of the function V, defined in Eqn. (8), with respect to the parameter vector θ is given by:

$$\nabla_\theta V(\psi, \theta) = \mathop{\mathbb{E}}_{\substack{s_0 \sim P_0(\cdot) \\ a_t \sim \pi_\theta(\cdot|s_t,t) \\ \xi_t \sim P_\xi(\cdot|s_t, a_t)}} \{ (\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t,t)) (\sum_{t=0}^{T-1} r_t) \} \ . \quad (13)$$

Definition 1. Let $(\mathcal{S}, \mathcal{A}, \Xi, P_0, f, \rho, P_\xi, T)$ and π be an environment and a policy, respectively, as defined in Section 2. We call a history h of the policy in the environment, the sequence:

$$h = (s_0, a_0, \xi_0, r_0, a_1, \xi_1, r_1, \dots a_{T-1}, \xi_{T-1}, r_{T-1}) \ , \quad (14)$$

where s_0 is an initial state sampled from P_0, and where, at time t, ξ_t is a disturbance sampled from P_ξ, a_t is an action sampled from π, and r_t is the reward observed.

For computing the gradients, our DESGA algorithm will exploit the following theorem that shows that an unbiased estimate of the gradients can be obtained by evaluating the gradients of a loss function computed from a set of histories. Automatic differentiation will later be used for computing these gradients in our simulations.

Theorem 2. Let $(\mathcal{S}, \mathcal{A}, \Xi, P_0, f_\psi, \rho_\psi, P_\xi, T)$ and π_θ be an environment and a policy, respectively, as defined in Section 2.2. Let $V(\psi, \theta)$ be the expected cumulative reward of policy π_θ averaged over the initial states, as defined in Eqn. (8). Let $\mathcal{D} = \{h^m | m = 0, \dots, M-1\}$ be a set of M histories sampled independently and identically from the policy π_θ in the environment. Let $\mathcal{L}$ be a

loss function such that, $\forall(\psi, \theta) \in \Psi \times \Theta$:

$$\mathcal{L}(\psi,\theta) = -\frac{1}{M}\sum_{m=0}^{M-1}\Big(\sum_{t=0}^{T-1}\log\pi_\theta(a_t^m|s_t^m,t) + \log P_\xi(\xi_t^m|s_t^m,a_t^m)\Big) \times \Big((\sum_{t=0}^{T-1} r_t^m) - B\Big) + \Big(\sum_{t=0}^{T-1}\rho_\psi(s_t^m,a_t^m,\xi_t^m)\Big)\Big) \ , \quad (15)$$

where B is a constant value called the baseline.

The gradients with respect to ψ and θ of the loss function are unbiased estimators of the gradients of the function V as defined in Eqn. (8) with opposite directions, i.e. they are such that:

$$\mathop{\mathbb{E}}_{\substack{s_0\sim P_0(\cdot)\\ a_t\sim\pi_\theta(\cdot|s_t,t)\\ \xi_t\sim P_\xi(\cdot|s_t,a_t)}}\{\nabla_\psi\mathcal{L}(\psi,\theta)\} = -\nabla_\psi V(\psi,\theta) \quad (16)$$

$$\mathop{\mathbb{E}}_{\substack{s_0\sim P_0(\cdot)\\ a_t\sim\pi_\theta(\cdot|s_t,t)\\ \xi_t\sim P_\xi(\cdot|s_t,a_t)}}\{\nabla_\theta\mathcal{L}(\psi,\theta)\} = -\nabla_\theta V(\psi,\theta) \ . \quad (17)$$

Corollary 4. The gradient of the loss function, defined in Eqn. 15, with respect to θ corresponds to the opposite of the update direction computed with the REINFORCE algorithm [Williams, 1992] averaged over M simulations.

The proofs for the theorems and corollaries presented in this section are given in Appendix A.

3.2 Parameter optimization with projected stochastic gradient ascent

In the previous section, we have developed an analytical expression for the computation of the gradients of the expected cumulative reward with respect to the parameters of the environment and of the policy. In order to allow for the event where these parameters belong to a constrained set, our DESGA algorithm will use the projected gradient ascent method [Cohen et al., 2016].

Gradient ascent is an optimization technique where the optimized variables are updated at each iteration step k, by a fixed-size step that is proportional to the gradient of the objective function with respect to these variables. The size of the update can be controlled by parameter α, called the learning rate. In the problem defined by Eqn. (7), we aim to find a parameter vector $x = (\psi, \theta) \in X = \Psi \times \Theta \subset \mathbb{R}^{d_\Psi + d_\Theta}$ that maximizes the expected cumulative reward. Gradient ascent updates the parameter vector x_k at time k as:

$$x_{k+1} \leftarrow x_k + \alpha \cdot \nabla_x V(x_k) \ . \quad (18)$$

The new point x_{k+1} computed by simple gradient ascent according to Eqn. (18), may not belong to the constraint set X. In projected gradient ascent, we choose the point nearest to x_{k+1}, according to the Euclidean distance, that is located in the set X i.e., the projection of x_{k+1} onto the set X. The projection Π_X of a point y onto a set X is defined as:

$$\Pi_X(y) = \arg\min_{x\in X}\frac{1}{2} \parallel x - y \parallel_2^2 \ . \quad (19)$$

Using projected gradient ascent, we first compute the update:

$$y_{k+1} \leftarrow x_k + \alpha \cdot \nabla_x V(x_k) \ , \quad (20)$$

and then we project the new point y_{k+1} into the feasible set X, according to:

$$x_{k+1} \leftarrow \Pi_X(y_{k+1}) \ . \quad (21)$$

The projected gradient descent (or ascent) shares the same convergence rate and guarantees as the unconstrained case, under specific conditions on the smoothness and the convexity of the objective function [Cohen et al., 2016]. However, the computational cost of the projection operation depends on the characteristics of the constrained space X. Let us also remark that, in practice, we assume the gradients to exist on the boundary of $\Psi \times \Theta$. If this assumption does not hold, we can consider a

compact subset K of the interior of $\Psi \times \Theta$ such that Theorem 1 ensures the existence of the gradients on K.

The DESGA algorithm will update the vector of parameters ψ and θ according to Eqns. (20) and (21). In practice, the gradients are approximated using Theorem 2, such that projected stochastic gradient ascent is performed. Furthermore, we choose as the baseline the expected cumulative reward approximated by averaging the observed cumulative reward over the M histories h^m used for computing the loss function:

$$B = \frac{1}{M} \sum_{m=0}^{M-1} \sum_{t=0}^{T-1} r_t^m \ . \tag{22}$$

The execution of projected stochastic gradient ascent algorithm for optimizing the objective in Eqn. (7) is fully detailed in Algorithm 1 in Appendix B.

4 Experiments

In this section, we first introduce the methodology used for assessing the performance of DESGA. Afterwards, we test the DESGA algorithm on two benchmarks, the Mass-Spring-Damper (MSD) environment and one related to the design of a solar off-grid microgrid. Both environments are fully described in Appendices C and D. [2]

4.1 Methodology

When running the DESGA algorithm on a test problem, we will report the following results. First, at every iteration k of the algorithm we will compute the expected return of the policy on the environment for the current pair of parameter vectors (θ_k, ψ_k), that is $V(\psi_k, \theta_k)$. This value is computed by running 100 Monte-Carlo simulations. Since the DESGA algorithm is stochastic, we will actually report the average of this value obtained over 20 runs (random seeds) of the algorithm. The standard deviation over the 20 runs of the algorithm will also be reported.

For every problem we will also compare the performance of DESGA with an algorithm based on a discretization Ψ_d of the environment's hypothesis space Ψ. This algorithm will run the REINFORCE algorithm for every value of $\psi_d \in \Psi_d$ and compute the expected return of the policy obtained using 100 Monte-Carlo simulations. The process will be repeated five times to estimate the average expected return that could be obtained by a policy learned by the REINFORCE algorithm for each ψ_d.

4.2 Mass-Spring-Damper environment

We consider here the MSD environment $(\mathcal{S}, \mathcal{A}, \Xi, P_0, f_\psi, \rho_\psi, P_\xi, T)$ described in detail in Appendix C.

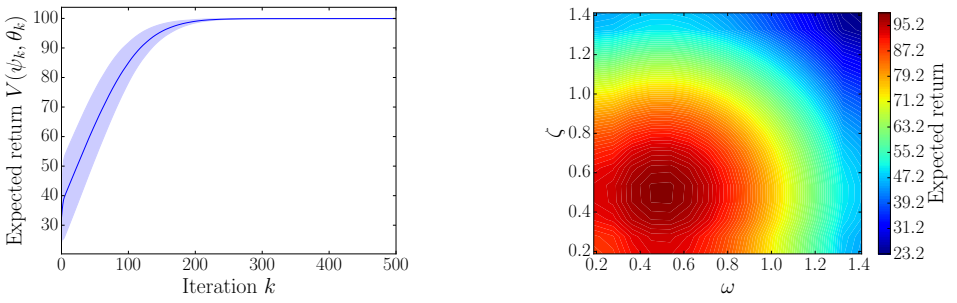
Hypothesis spaces. The environment is parametrized by the real vector $\psi = (\omega, \zeta, \phi_0, \phi_1, \phi_2) \in \Psi = [0.1, 1.5] \times [0.1, 1.5] \times [-2, 2] \times [-2, 2] \times [-2, 2] \subset \mathbb{R}^5$. We will constrain the hypothesis space for the policies to time-invariant policies, meaning $\pi_\theta(a|s,t) = \pi_\theta(a|s,t'), \forall a \in \mathcal{A}, \forall s \in \mathcal{S}, \forall t, t' \in \{0, \dots, T-1\}$. Any of these policies is a multi-layer perceptron (MLP) with two inputs (one for each value of the state vector s), and with one hidden layer of 128 neurons with hyperbolic tangent activation functions. The MLP has five output neurons ($|\mathcal{A}| = 5$) from which a probability distribution over $\mathcal{A}$ will be inferred using a softmax function. All the possible values for the parameters of the MLP define the policy's hypothesis space Θ.

Parameters of the DESGA algorithm. The gradients are evaluated applying automatic differentiation on the loss function defined in Eqn. (15). Furthermore, the *Adam* algorithm is used for updating (ψ, θ). It is a variant on the vanilla stochastic gradient ascent given in Algorithm 1 which has proven to perform well on highly non-convex problems [Kingma and Ba, 2014]. The gradients are estimated

[2] The implementation of our algorithm and of the different benchmarks are provided in the following github repository:
`https://github.com/adrienBolland/Direct-Environment-Search-with-Gradient-Ascent`

on batches of $M = 64$ trajectories and the stepsize α of the Adam algorithm is chosen equal to 0.005. We keep the default values for the other parameters of the Adam algorithm. Furthermore, the states are z-normalized by an average vector corresponding to the equilibrium position $(x_{eq}, 0)$ targeted by an optimal policy, as explained in Appendix C. The standard deviation of the scaling is chosen equal to $(0.005, 0.02)$, an approximation of the standard deviation vector of the states collected over high-performing trajectories.

Performance of the DESGA algorithm. Figure 1a shows the evolution of the expected return, estimated with 100 Monte-Carlo samples, averaged over 20 runs of the DESGA algorithm. The standard deviation between the different runs is illustrated by the shaded area around the mean. As we can see, the DESGA algorithm converges towards a maximal expected return almost equal to 100. We note that 100 is an upper-bound on the return that can only be reached if at each time-step t, the position of the mass is at its equilibrium x_{eq}. The standard deviation also strongly decreases as the iterations go on. We discovered that by using time-variant policies, better results could not be obtained for this problem. Furthermore, Fig 1b shows the average expected return of 5 policies computed by the REINFORCE algorithm for each $\psi_d \in \Psi_d = \Omega_d \times Z_d \times \{c_0\} \times \{c_1\} \times \{c_2\}$ where $\Omega_d = Z_d = \{0.1 + k \cdot \Delta | k = 1, \ldots, 15\}$ with $\Delta = 0.082$. We note that, c_0, c_1 and c_2 correspond to an optimal triplet of values for ϕ_1, ϕ_2 and ϕ_3, respectively, as described in Appendix C. The highest average expected return of the policies occurs for $(\omega, \zeta) = (0.5, 0.5)$. Finally, the average expected return of the policies identified by the REINFORCE algorithm, for this value of $\psi = (0.5, 0.5, c_0, c_1, c_2)$, was almost identical to the expected returns obtained by the policies computed with the DESGA algorithm. We also note that the DESGA algorithm converged at every run towards a ψ whose ω and ζ components were both equal to 0.5 and whose triplet (ϕ_0, ϕ_1, ϕ_2) was always optimal, but not necessarily equal to (c_0, c_1, c_2).



(a) Evolution of the expected return

(b) Performance of REINFORCE on Ψ_d

Figure 1: Assessment of DESGA. Left: the average value of $V(\theta_k, \psi_k)$ and its standard deviation over 20 executions of the DESGA algorithm as a function of the number of iterations k of the algorithm. Right: the average expected return of five policies identified with the REINFORCE algorithm for every element $\psi_d \in \Psi_d$.

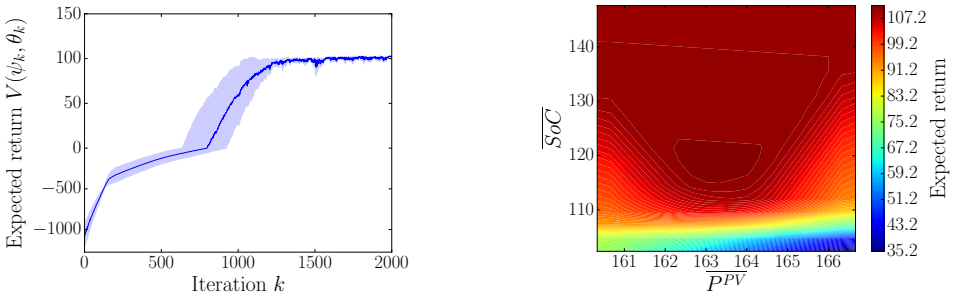
4.3 Sizing and operation of a solar off-grid microgrid

In this section, we consider the solar off-grid microgrid environment $(\mathcal{S}, \mathcal{A}, \Xi, P_0, f_\psi, \rho_\psi, P_\xi, T)$ presented in Appendix D.

Hypothesis spaces. The environment is parametrized by the real vector $\psi = (\overline{SoC}, \overline{P^{PV}}) \in \Psi = [0, 200] \times [0, 300]$. We will constrain the hypothesis space for the policies to time-invariant Gaussian policies, i.e. policies such that $\pi_\theta(a|s,t) = \mathcal{N}(a|\mu_\theta(s), \sigma_\theta(s)),\ \forall a \in \mathcal{A}, \forall s \in \mathcal{S}, \forall t \in \{0, \ldots, T-1\}$ where $\mu_\theta(s)$ and $\sigma_\theta(s)$ are the expectation and the standard deviation of the normal distribution $\mathcal{N}$ in function of the state s and of the parameter vector θ, respectively. A MLP with four inputs (one for each value of the state vector s), and with one hidden layer of 128 neurons with hyperbolic tangent activation functions, outputs the two values $\mu_\theta(s)$ and $\sigma_\theta(s)$. All the possible values for the parameters of the MLP define the policy's hypothesis space Θ.

Parameters of the DESGA algorithm. The parameters related to the optimization process are the same as those used for the MSD environment in Section 4.2. The states are z-normalized by the average vector $(100, 12, 6.31, 6.48)$ and by the standard deviation vector $(50, 6, 8.9, 2)$. These values represent the mean and the standard deviation of the state vector for a microgrid configuration where $\psi = (200, 300)$. The rewards collected are scaled linearly from the interval $[-5000, 0]$ to the interval $[0, 1]$. Moreover, the vector ψ is scaled from $[0, 200] \times [0, 300]$ to $[0, 1] \times [0, 1]$ in the interest of keeping the optimization variables in a small range.

Performance of the DESGA algorithm. Similar to Section 4.2, Figure 2a presents the evolution of the average expected (scaled) return collected in the solar off-grid microgrid environment, averaged over 20 runs of the DESGA algorithm. As we can see, the DESGA algorithm converges towards a maximal expected return that stands around a value of 100. We note that 120 is an upper bound on the expected return that can only be reached if, during the entire horizon ($T = 120$), the instantaneous reward takes the value one. The standard deviation also strongly decreases as the iterations go on. Furthermore, Fig 2b shows the average expected return of five policies computed with the REINFORCE algorithm at each point $\psi^d \in \Psi^d = \{0, 2, ..., 200\} \times \{0, 3, ..., 300\}$, where Ψ^d is a discrete subset of the hypothesis space Ψ that forms a mesh 100×100. We also note that the DESGA algorithm is converging at every run towards a value of $\psi = (\overline{SoC}, \overline{P^{PV}}) = (114, 165)$, which is very close to the value of $\psi^d = (114, 166)$ that leads to the highest average expected return of policies computed with the REINFORCE algorithm.



(a) Evolution of the expected return

(b) Performance of REINFORCE on Ψ_d

Figure 2: Assessment of DESGA on the solar off-grid microgrid environment. Left: the average value of $V(\theta_k, \psi_k)$ and its standard deviation over 20 executions of the DESGA algorithm as a function of the number of iterations k of the algorithm. Right: the average expected return of five policies identified with the REINFORCE algorithm for every element $\psi_d \in \Psi_d$. A magnified area of the original graph is presented, where the maximum values are located.

5 Conclusions and Future Work

In this paper, we propose an algorithm that can jointly optimize an RL environment and a policy with maximal expected return over a joint hypothesis space of environments and policies. This algorithm is suited to cases in which the design of the environment and the applied policy are interdependent. We demonstrate the performance of DESGA on the design of an MSD environment and on the sizing of an autonomous energy system. The results show that the DESGA algorithm outputs a solution which is equivalent in terms of performances to the one obtained by the REINFORCE algorithm run for every element of a finely discretized environment's hypothesis space.

In this paper, the DESGA algorithm was designed in the context of jointly optimizing the design of a discrete-time dynamical system and its policy. This algorithm could be extended to the case where the environment is a finite-time Markov Decision Process (MDP) performing a similar development as the one presented in Section 3.1. The approach could also be extended to environments with infinite-time horizons.

Future work could also be directed on an approximation of the gradients. With the computational complexity of the automatic differentiation being proportional to the optimization horizon, the

problem may become intractable for long horizons. An analytical bound on the error when performing this approximation would be valuable for striking a trade-off between computational efficiency and the quality of the solution.

Additionally, as future work, the proposed method could also be combined with recent research in gradient-based direct policy search. The use of actor-critic methods, proximal policy optimization, etc., that are shown to result in stable learning and efficient exploration, could lead to better performances. This would come at the expense of involving the additional approximation architecture (set of parameters) of a value function.

Finally, in this paper we assumed that we have direct access to the parametrized dynamics of the system, the reward function, and the disturbance function. In the event these assumptions do not hold, we propose constructing an approximation of these functions by a differentiable function approximator as future work. This would introduce an additional learning step, in order to obtain a good approximation architecture from observations, which would then be used in the proposed algorithm.

References

[Ajwad et al., 2018] Ajwad, S. A., Iqbal, J., Islam, R. U., Alsheikhy, A., Almeshal, A., and Mehmood, A. (2018). Optimal and robust control of multi dof robotic manipulator: Design and hardware realization. *Cybernetics and Systems*, 49(1):77–93.

[Bertsekas, 2005] Bertsekas, D. P. (2005). *Dynamic programming and optimal control*, volume 1. Athena scientific Belmont, MA.

[Boukas et al., 2020] Boukas, I., Ernst, D., Théate, T., Bolland, A., Huynen, A., Buchwald, M., Wynants, C., and Cornélusse, B. (2020). A deep reinforcement learning framework for continuous intraday market bidding. *arXiv preprint arXiv:2004.05940*.

[Buşoniu et al., 2011] Buşoniu, L., Ernst, D., De Schutter, B., and Babuška, R. (2011). Cross-entropy optimization of control policies with adaptive basis functions. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 41(1):196–209.

[Castejón et al., 2010] Castejón, C., Carbone, G., García Prada, J., and Ceccarelli, M. (2010). A multi-objective optimization of a robotic arm for service tasks. *Strojniski Vestnik/Journal of Mechanical Engineering*, 56(5).

[Cohen et al., 2016] Cohen, K., Nedic, A., and Srikant, R. (2016). On projected stochastic gradient descent algorithm with weighted averaging for least squares regression. *arXiv preprint arXiv:1606.03000*.

[François-Lavet et al., 2016] François-Lavet, V., Gemine, Q., Ernst, D., and Fonteneau, R. (2016). 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.

[Hansen and Ostermeier, 1996] Hansen, N. and Ostermeier, A. (1996). Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation. In *Proceedings of IEEE International Conference on Evolutionary Computation*, pages 312–317.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Schulman et al., 2015] Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015). Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897.

[Szita and Lörincz, 2006] Szita, I. and Lörincz, A. (2006). Learning tetris using the noisy cross-entropy method. *Neural computation*, 18(12):2936–2941.

[Williams, 1992] Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Appendices

A Analytical derivation of the gradient for learning optimal environments

Theorem 1. Let $(\mathcal{S}, \mathcal{A}, \Xi, P_0, f_\psi, \rho_\psi, P_\xi, T)$ and π_θ be an environment and a policy as defined in Section 2.2. Additionally, let the functions f_ψ, ρ_ψ and P_ξ be continuously differentiable over their domain of definition. Let $V(\psi, \theta)$ be the expected cumulative reward of policy π_θ, averaged over the initial states, for all $(\psi, \theta) \in \Psi \times \Theta$ as defined in Eqn. (8).

Then, the function V exists, is bounded, and is continuously differentiable in the interior of $\Psi \times \Theta$.

Proof. Let us first define the random variable associating the cumulative reward to a realization of a trajectory sampled from a policy in the environment for fixed parameter vectors $(\psi, \theta) \in \Psi \times \Theta$. We prove its expectation exists and is bounded for all $(\psi, \theta) \in \Psi \times \Theta$. Furthermore, $V(\psi, \theta)$ is defined by a parametric integral which we prove to be continuously differentiable for all $(\psi, \theta) \in \Psi \times \Theta$.

Let $\mathcal{R}_{\psi,\theta}$ be the real random variable that associates to the realization of a trajectory given $\psi \in \Psi$ and $\theta \in \Theta$ its cumulative reward . Given a trajectory τ, the random variable $\mathcal{R}_{\psi,\theta}$ takes as values $R_{\psi,\theta}(\tau)$ as defined in Eqn. (4). Let $P_{\mathcal{R}_{\psi,\theta}}$ be the induced probability of this random variable. We can write:

$$P_{\mathcal{R}_{\psi,\theta}} = P_{\psi,\theta}(s_0, a_0, \xi_0, a_1, \xi_1, \dots, a_{T-1}, \xi_{T-1}) \tag{23}$$

$$= P_0(s_0) \prod_{t=0}^{T-1} \pi_\theta(a_t|s_t, t) P_\xi(\xi_t|s_t, a_t) \ , \tag{24}$$

where $s_{t+1} = f_\psi(s_t, a_t, \xi_t)$. The expected cumulative reward given in Eqn. (8) is the expectation of the random variables $\mathcal{R}_{\psi,\theta}$. If the expectation exists, it can therefore be written as:

$$V(\psi, \theta) = \int \big(P_0(s_0) \prod_{t=0}^{T-1} \pi_\theta(a_t|s_t, t) P_\xi(\xi_t|s_t, a_t)\big) \big(\sum_{t=0}^{T-1} \rho_\psi(s_t, a_t, \xi_t)\big) \, ds_0 da_0 \dots da_{T-1} d\xi_0 \dots d\xi_{T-1} \ , \tag{25}$$

or, more simply, as:

$$V(\psi, \theta) = \int P_{\mathcal{R}_{\psi,\theta}}(\tau) R_{\psi,\theta}(\tau) d\tau \ . \tag{26}$$

The integration theory has shown that a measurable function upper-bounded in norm almost-everywhere by an integrable function on a domain is itself integrable on this domain. Moreover, a random variable is measurable by definition and the cumulative reward is such that:

$$\int |P_{\mathcal{R}_{\psi,\theta}} R_{\psi,\theta}(\tau)| d\tau \leq \int P_{\mathcal{R}_{\psi,\theta}} \, T \, r_{max} d\tau \leq T \, r_{max} \ . \tag{27}$$

The integral defined by Eqn. (26) thus exists and the function V is bounded for all $(\psi, \theta) \in \Psi \times \Theta$.

As a corollary to the Leibniz integral rule, a function defined as in Eqn. (26) is continuously differentiable on the interior of the set $\Psi \times \Theta$ if $P_{\mathcal{R}_{\psi,\theta}} R_{\psi,\theta}(\tau)$ is continuously differentiable on the compact $\Psi \times \Theta \times X$ where $X = \mathcal{S} \times (\mathcal{A} \times \Xi)^T$ is the set of all trajectories. The latter is true by hypothesis. Furthermore, it implies that the partial derivative of the integral equals the integral of the partial derivative of the integrand.

□

Corollary 1. The function V, as defined in Theorem 1, exists, is bounded and is continuously differentiable on the interior of $\Psi \times \Theta$ if $\mathcal{A}$ and/or Ξ are discrete.

Proof. Let us write the expression of the expectation (8) in the three cases depending on whether $\mathcal{A}$ and/or Ξ are discrete and show that the different results of Theorem 1 are still valid.

1. If $\mathcal{A}$ is discrete:

$$V(\psi,\theta) = \int \sum_{(a_0,\dots a_{T-1})\in\mathcal{A}^T} \left(P_0(s_0)\prod_{t=0}^{T-1}\pi_\theta(a_t|s_t,t)P_\xi(\xi_t|s_t,a_t)\right) \left(\sum_{t=0}^{T-1}\rho_\psi(s_t,a_t,\xi_t)\right) ds_0 d\xi_0\dots d\xi_{T-1} \,. \quad (28)$$

2. If Ξ is discrete:

$$V(\psi,\theta) = \int \sum_{(\xi_0,\dots \xi_{T-1})\in\Xi^T} \left(P_0(s_0)\prod_{t=0}^{T-1}\pi_\theta(a_t|s_t,t)P_\xi(\xi_t|s_t,a_t)\right) \left(\sum_{t=0}^{T-1}\rho_\psi(s_t,a_t,\xi_t)\right) ds_0 da_0\dots da_{T-1} \,. \quad (29)$$

3. If $\mathcal{A}$ and Ξ are discrete:

$$V(\psi,\theta) = \int \sum_{(a_0,\dots a_{T-1})\in\mathcal{A}^T}\sum_{(\xi_0,\dots \xi_{T-1})\in\Xi^T} \left(P_0(s_0)\prod_{t=0}^{T-1}\pi_\theta(a_t|s_t,t)P_\xi(\xi_t|s_t,a_t)\right) \left(\sum_{t=0}^{T-1}\rho_\psi(s_t,a_t,\xi_t)\right) ds_0 \,. \quad (30)$$

In the three cases, we can still bound the integral as in Eqn. (27) and apply the corollary of the Leibniz integral rule if the integrand is continuously differentiable for all discrete values. Finally, by linearity of the differential operator, the operator can be distributed on the terms of the different sums when computing the derivative of the function V.

□

Corollary 2. The gradient of the function V defined in Eqn. (8) with respect to the parameter vector ψ is such that:

$$\nabla_\psi V(\psi,\theta) = \mathop{\mathbb{E}}_{\substack{s_0\sim P_0(\cdot)\\ a_t\sim\pi_\theta(\cdot|s,t)\\ \xi_t\sim P_\xi(\cdot|s_t,a_t)}} \left\{\left(\sum_{t=0}^{T-1}\left(\nabla_s\log\pi_\theta(a_t|s,t)|_{s=s_t} + \nabla_s\log P_\xi(\xi_t|s,a_t)|_{s=s_t}\right)\cdot\nabla_\psi s_t\right) \times\left(\sum_{t=0}^{T-1} r_t\right) + \left(\sum_{t=0}^{T-1}\nabla_\psi\rho_\psi(s,a_t,\xi_t)|_{s=s_t} + \nabla_s\rho_\psi(s,a_t,\xi_t)|_{s=s_t}\cdot\nabla_\psi s_t\right\}, \quad (31)$$

where:

$$\nabla_\psi s_t = (\nabla_s f_\psi)(s,a_{t-1},\xi_{t-1})|_{s=s_{t-1}}\cdot\nabla_\psi s_{t-1} + (\nabla_\psi f_\psi)(s,a_{t-1},\xi_{t-1})|_{s=s_{t-1}} \,, \quad (32)$$

with $\nabla_\psi s_0 = 0$.

Proof. To compute this gradient, we first apply the product rule for gradients to Eqn. (8). Afterwards, we exploit the equality $\nabla f = f\nabla\log f$ that holds if f is a continuously differentiable function.

$$\nabla_\psi V(\psi,\theta) = \int(\nabla_\psi P_{\mathcal{R}_{\psi,\theta}}(\tau))R_{\psi,\theta}(\tau)d\tau + \int P_{\mathcal{R}_{\psi,\theta}}(\tau)(\nabla_\psi R_{\psi,\theta}(\tau))d\tau \quad (33)$$

$$= \int P_{\mathcal{R}_{\psi,\theta}}(\tau)(\nabla_\psi\log P_{\mathcal{R}_{\psi,\theta}}(\tau))R_{\psi,\theta}(\tau)d\tau + \int P_{\mathcal{R}_{\psi,\theta}}(\tau)(\nabla_\psi R_{\psi,\theta}(\tau))d\tau \quad (34)$$

$$= \mathop{\mathbb{E}}_{\substack{s_0\sim P_0(\cdot)\\ a_t\sim\pi_\theta(\cdot|s_t,t)\\ \xi_t\sim P_\xi(\cdot|s_t,a_t)}} \left\{(\nabla_\psi\log P_{\mathcal{R}_{\psi,\theta}}(\tau))R_{\psi,\theta}(\tau) + (\nabla_\psi R_{\psi,\theta}(\tau))\right\}. \quad (35)$$

By applying the logarithmic operator to both sides of Eqn. (24), we have:

$$\log P_{\mathcal{R}_{\psi,\theta}}(\tau) = \log P_0(s_0) + \sum_{t=0}^{T-1} \log \pi_\theta(a_t|s_t,t) + \sum_{t=0}^{T-1} \log P_\xi(\xi_t|s_t,a_t) \ . \tag{36}$$

Let $\cdot$ denote the dot product operator. Using the chain rule formula together with Eqn. (4), we can write:

$$\nabla_\psi \log \pi_\theta(a_t|s_t,t) = \nabla_s \log \pi_\theta(a_t|s,t)|_{s=s_t} \cdot \nabla_\psi s_t \tag{37}$$

$$\nabla_\psi \log P_\xi(\xi_t|s_t,a_t) = \nabla_s \log P_\xi(\xi_t|s,a_t)|_{s=s_t} \cdot \nabla_\psi s_t \tag{38}$$

$$\nabla_\psi \rho_\psi(s_t,a_t,\xi_t) = \nabla_\psi \rho_\psi(s,a_t,\xi_t)|_{s=s_t} + \nabla_s \rho_\psi(s,a_t,\xi_t)|_{s=s_t} \cdot \nabla_\psi s_t \ , \tag{39}$$

where:

$$\nabla_\psi s_t = (\nabla_s f_\psi)(s,a_{t-1},\xi_{t-1})|_{s=s_{t-1}} \cdot \nabla_\psi s_{t-1} + (\nabla_\psi f_\psi)(s,a_{t-1},\xi_{t-1})|_{s=s_{t-1}} \ , \tag{40}$$

with $\nabla_\psi s_0 = 0$.

Finally, combining the previous results with Eqns. (35) and (36), we have:

$$\nabla_\psi V(\psi,\theta) = \mathop{\mathbb{E}}_{\substack{s_0\sim P_0(\cdot) \\ a_t\sim\pi_\theta(\cdot|s,t) \\ \xi_t\sim P_\xi(\cdot|s_t,a_t)}} \Big\{ \Big(\sum_{t=0}^{T-1} \big(\nabla_s \log \pi_\theta(a_t|s,t)|_{s=s_t} + \nabla_s \log P_\xi(\xi_t|s,a_t)|_{s=s_t} \big) \cdot \nabla_\psi s_t \Big)$$
$$\times \Big(\sum_{t=0}^{T-1} r_t \Big) + \Big(\sum_{t=0}^{T-1} \nabla_\psi \rho_\psi(s_t,a_t,\xi_t) \Big) \Big\} \ . \tag{41}$$

□

Corollary 3. The gradient of the function V defined in Eqn. (8) with respect to the parameter vector θ is given by:

$$\nabla_\theta V(\psi,\theta) = \mathop{\mathbb{E}}_{\substack{s_0\sim P_0(\cdot) \\ a_t\sim\pi_\theta(\cdot|s_t,t) \\ \xi_t\sim P_\xi(\cdot|s_t,a_t)}} \{ (\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t,t)) (\sum_{t=0}^{T-1} r_t) \} \ . \tag{42}$$

Proof. Using similar derivations as for the Corollary 1, we have for the gradient with respect to θ:

$$\nabla_\theta V(\psi,\theta) = \mathop{\mathbb{E}}_{\substack{s_0\sim P_0(\cdot) \\ a_t\sim\pi_\theta(\cdot|s_t,t) \\ \xi_t\sim P_\xi(\cdot|s_t,a_t)}} \{ (\nabla_\theta \log P_{\mathcal{R}_{\psi,\theta}}(\tau)) R_{\psi,\theta}(\tau) \} \tag{43}$$

$$= \mathop{\mathbb{E}}_{\substack{s_0\sim P_0(\cdot) \\ a_t\sim\pi_\theta(\cdot|s_t,t) \\ \xi_t\sim P_\xi(\cdot|s_t,a_t)}} \{ (\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t,t)) (\sum_{t=0}^{T-1} r_t) \} \ . \tag{44}$$

□

Theorem 2. Let $(\mathcal{S},\mathcal{A},\Xi,P_0,f_\psi,\rho_\psi,P_\xi,T)$ and π_θ be an environment and a policy as defined in Section 2.2. Let $V(\psi,\theta)$ be the expected cumulative reward of policy π_θ averaged over the initial states as defined in Eqn. (8). Let $\mathcal{D} = \{h^m | m = 0,\ldots,M-1\}$ be a set of M histories sampled independently and identically from the policy π_θ in the environment. Let $\mathcal{L}$ be a loss function such that, $\forall(\psi,\theta) \in \Psi \times \Theta$:

$$\mathcal{L}(\psi,\theta) = -\frac{1}{M} \sum_{m=0}^{M-1} \Big(\sum_{t=0}^{T-1} \log \pi_\theta(a_t^m|s_t^m,t) + \log P_\xi(\xi_t^m|s_t^m,a_t^m) \Big)$$
$$\times \big((\sum_{t=0}^{T-1} r_t^m) - B \big) + \big(\sum_{t=0}^{T-1} \rho_\psi(s_t^m,a_t^m,\xi_t^m) \big) \Big) \ , \tag{45}$$

where B is a constant value called the baseline.

The gradients with respect to ψ and θ of the loss function are unbiased estimators of the gradients of the function V as defined in Eqn. (8), with opposite directions, i.e. such that:

$$\mathop{\mathbb{E}}_{\substack{s_0\sim P_0(\cdot)\\ a_t\sim\pi_\theta(\cdot|s_t,t)\\ \xi_t\sim P_\xi(\cdot|s_t,a_t)}}\{\nabla_\psi\mathcal{L}(\psi,\theta)\} = -\nabla_\psi V(\psi,\theta) \tag{46}$$

$$\mathop{\mathbb{E}}_{\substack{s_0\sim P_0(\cdot)\\ a_t\sim\pi_\theta(\cdot|s_t,t)\\ \xi_t\sim P_\xi(\cdot|s_t,a_t)}}\{\nabla_\theta\mathcal{L}(\psi,\theta)\} = -\nabla_\theta V(\psi,\theta)\ . \tag{47}$$

Proof. Let us first rewrite the loss function using the notations of Theorem 1. We have:

$$\mathcal{L}(\psi,\theta) = -\frac{1}{M}\sum_{m=0}^{M-1}(\log P_{\mathcal{R}_{\psi,\theta}}(\tau^m) - \log P_0(s_0^m))\big((\sum_{t=0}^{T-1} r_t^m) - B\big) + (R_{\psi,\theta}(\tau^m))\ . \tag{48}$$

The expectation of the gradient with respect to ψ is given by:

$$\begin{aligned}\mathop{\mathbb{E}}_{\substack{s_0\sim P_0(\cdot)\\ a_t\sim\pi_\theta(\cdot|s_t,t)\\ \xi_t\sim P_\xi(\cdot|s_t,a_t)}}\{\nabla_\psi\mathcal{L}(\psi,\theta)\} = \mathop{\mathbb{E}}_{\substack{s_0\sim P_0(\cdot)\\ a_t\sim\pi_\theta(\cdot|s_t,t)\\ \xi_t\sim P_\xi(\cdot|s_t,a_t)}}\{-\frac{1}{M}\sum_{m=0}^{M-1}\nabla_\psi(\log P_{\mathcal{R}_{\psi,\theta}}(\tau^m) - \log P_0(s_0^m))\\ \times\big((\sum_{t=0}^{T-1} r_t^m) - B\big) + \nabla_\psi(R_{\psi,\theta}(\tau^m))\}\ .\end{aligned} \tag{49}$$

Observing that every term in the sum has the same expectation and that $\nabla_\psi \log P_0(s_0^m) = 0$, we can write:

$$\mathop{\mathbb{E}}_{\substack{s_0\sim P_0(\cdot)\\ a_t\sim\pi_\theta(\cdot|s_t,t)\\ \xi_t\sim P_\xi(\cdot|s_t,a_t)}}\{\nabla_\psi\mathcal{L}(\psi,\theta)\} = -\mathop{\mathbb{E}}_{\substack{s_0\sim P_0(\cdot)\\ a_t\sim\pi_\theta(\cdot|s_t,t)\\ \xi_t\sim P_\xi(\cdot|s_t,a_t)}}\{\nabla_\psi(\log P_{\mathcal{R}_{\psi,\theta}}(\tau))\times\big((\sum_{t=0}^{T-1} r_t) - B\big) + \nabla_\psi(R_{\psi,\theta}(\tau))\}\ . \tag{50}$$

Moreover:

$$\mathop{\mathbb{E}}_{\substack{s_0\sim P_0(\cdot)\\ a_t\sim\pi_\theta(\cdot|s_t,t)\\ \xi_t\sim P_\xi(\cdot|s_t,a_t)}}\{\nabla_\psi(\log P_{\mathcal{R}_{\psi,\theta}}(\tau))B\} = \nabla_\psi \mathop{\mathbb{E}}_{\substack{s_0\sim P_0(\cdot)\\ a_t\sim\pi_\theta(\cdot|s_t,t)\\ \xi_t\sim P_\xi(\cdot|s_t,a_t)}}\{B\} = 0\ , \tag{51}$$

such that:

$$\mathop{\mathbb{E}}_{\substack{s_0\sim P_0(\cdot)\\ a_t\sim\pi_\theta(\cdot|s_t,t)\\ \xi_t\sim P_\xi(\cdot|s_t,a_t)}}\{\nabla_\psi\mathcal{L}(\psi,\theta)\} = -\nabla_\psi V(\psi,\theta)\ . \tag{52}$$

Equivalently, the expectation of the gradient with respect to θ is given by:

$$\begin{aligned}\mathop{\mathbb{E}}_{\substack{s_0\sim P_0(\cdot)\\ a_t\sim\pi_\theta(\cdot|s_t,t)\\ \xi_t\sim P_\xi(\cdot|s_t,a_t)}}\{\nabla_\theta\mathcal{L}(\psi,\theta)\} = -\mathop{\mathbb{E}}_{\substack{s_0\sim P_0(\cdot)\\ a_t\sim\pi_\theta(\cdot|s_t,t)\\ \xi_t\sim P_\xi(\cdot|s_t,a_t)}}\{\nabla_\theta(\log P_{\mathcal{R}_{\psi,\theta}}(\tau))\\ \times\big((\sum_{t=0}^{T-1} r_t) - B\big) + \nabla_\theta(R_{\psi,\theta}(\tau))\}\ .\end{aligned} \tag{53}$$

The expectation of the term relative to the baseline is zero:

$$\mathop{\mathbb{E}}_{\substack{s_0\sim P_0(\cdot)\\ a_t\sim\pi_\theta(\cdot|s_t,t)\\ \xi_t\sim P_\xi(\cdot|s_t,a_t)}}\{\nabla_\theta(\log P_{\mathcal{R}_{\psi,\theta}}(\tau))B\} = \nabla_\theta \mathop{\mathbb{E}}_{\substack{s_0\sim P_0(\cdot)\\ a_t\sim\pi_\theta(\cdot|s_t,t)\\ \xi_t\sim P_\xi(\cdot|s_t,a_t)}}\{B\} = 0\ . \tag{54}$$

Furthermore, the gradient of the reward function with respect to θ is zero:

$$\mathop{\mathbb{E}}_{\substack{s_0\sim P_0(\cdot)\\ a_t\sim\pi_\theta(\cdot|s_t,t)\\ \xi_t\sim P_\xi(\cdot|s_t,a_t)}}\{\nabla_\theta(R_{\psi,\theta}(\tau))\} = \mathop{\mathbb{E}}_{\substack{s_0\sim P_0(\cdot)\\ a_t\sim\pi_\theta(\cdot|s_t,t)\\ \xi_t\sim P_\xi(\cdot|s_t,a_t)}}\{(\sum_{t=0}^{T-1}\nabla_\theta\rho_\psi(s_t,a_t,\xi_t)\} = 0\ . \tag{55}$$

We thus have that:

$$\mathop{\mathbb{E}}_{\substack{s_0\sim P_0(\cdot)\\ a_t\sim\pi_\theta(\cdot|s_t,t)\\ \xi_t\sim P_\xi(\cdot|s_t,a_t)}}\{\nabla_\theta\mathcal{L}(\psi,\theta)\} = -\mathop{\mathbb{E}}_{\substack{s_0\sim P_0(\cdot)\\ a_t\sim\pi_\theta(\cdot|s_t,t)\\ \xi_t\sim P_\xi(\cdot|s_t,a_t)}}\{\nabla_\theta(\log P_{\mathcal{R}_{\psi,\theta}}(\tau))\times\big(\sum_{t=0}^{T-1}r_t\big)\} \tag{56}$$

$$= -\mathop{\mathbb{E}}_{\substack{s_0\sim P_0(\cdot)\\ a_t\sim\pi_\theta(\cdot|s_t,t)\\ \xi_t\sim P_\xi(\cdot|s_t,a_t)}}\{\nabla_\theta(\sum_{t=0}^{T-1}\log\pi_\theta(a_t|s_t,t))\times\big(\sum_{t=0}^{T-1}r_t\big)\}\ . \tag{57}$$

Finally, we have that:

$$\mathop{\mathbb{E}}_{\substack{s_0\sim P_0(\cdot)\\ a_t\sim\pi_\theta(\cdot|s_t,t)\\ \xi_t\sim P_\xi(\cdot|s_t,a_t)}}\{\nabla_\theta\mathcal{L}(\psi,\theta)\} = -\nabla_\theta V(\psi,\theta)\ . \tag{58}$$

□

Corollary 4. The gradient of the loss function, defined in Eqn. 15, with respect to θ corresponds to the opposite of the update direction computed with the REINFORCE algorithm [Williams, 1992] averaged over M simulations.

Proof. The gradient of the loss function with respect to θ is given by:

$$\nabla_\theta\mathcal{L}(\psi,\theta) = -\sum_{m=0}^{M-1}\Big(\nabla_\theta(\log P_{\mathcal{R}_{\psi,\theta}}(\tau^m))\times\big(R_{\psi,\theta}(\tau^m)-B\big)\Big)\ . \tag{59}$$

The gradient is the opposite of the average over M trajectories of the update direction of the REINFORCE algorithm [Williams, 1992].

□

B Direct environment search with (projected stochastic) gradient ascent

Algorithm 1 DESGA

function `Optimize`$((\mathcal{S}, \mathcal{A}, \Xi, P_0, f_\psi, \rho_\psi, P_\xi, T), \pi_\theta, \Pi_\Psi, \Pi_\Theta)$
Parameter Number of gradient steps N
Parameter Batch size M
Parameter Learning rate α
for all $n \in \{0, \dots, N-1\}$ **do**
  for all $m \in \{0, \dots, M-1\}$ **do**
    $h =$ `GenerateHistory`$((\mathcal{S}, \mathcal{A}, \Xi, P_0, f_\psi, \rho_\psi, P_\xi, T), \pi_\theta)$
    Add h to the set $\mathcal{D}$
  end for
  Compute the baseline using the histories $B = \frac{1}{m} \sum_{m=0}^{M-1} \sum_{t=0}^{T-1} r_t$
  Differentiate Eqn. (15) for estimating the gradients Eqns. (11) and (13) using $\mathcal{D}$
  $(\psi, \theta) =$ `VanillaGradientAscent`$(\psi, \theta, \alpha, \hat{\nabla}_\psi V(\psi, \theta), \hat{\nabla}_\theta V(\psi, \theta))$
  $\psi \leftarrow \Pi_\Psi(\psi)$
  $\theta \leftarrow \Pi_\Theta(\theta)$
end for
return (ψ, θ)

function `GenerateHistory`$((\mathcal{S}, \mathcal{A}, \Xi, P_0, f_\psi, \rho_\psi, P_\xi, T), \pi_\theta)$
Sample an initial state: $s_0 \sim P_0(\cdot)$
for all $t \in \{0, \dots, T-1\}$ **do**
  $a_t \sim \pi_\theta(\cdot|s_t, t)$
  $\xi_t \sim P_\xi(\cdot|s_t, a_t)$
  $s_{t+1} = f_\psi(s_t, a_t, \xi_t)$
  $r_t = \rho_\psi(s_t, a_t, \xi_t)$
end for
$h = (s_0, a_0, \xi_0, r_0, a_1, \xi_1, \dots, a_{T-1}, \xi_{T-1}, r_{T-1})$
return h

function `VanillaGradientAscent`$(\psi, \theta, \alpha, \hat{\nabla}_\psi V(\psi, \theta), \hat{\nabla}_\theta V(\psi, \theta))$
$\psi \leftarrow \psi + \alpha \cdot \hat{\nabla}_\psi V(\psi, \theta)$
$\theta \leftarrow \theta + \alpha \cdot \hat{\nabla}_\theta V(\psi, \theta)$
return (ψ, θ)

C Mass-Spring-Damper environment

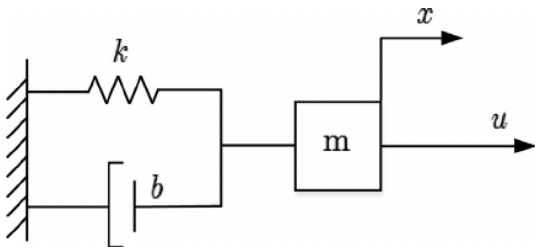


Figure 3: Mass-Spring-Damper system.

Let us consider a Mass-Spring-Damper (MSD) system defined as follows. A point mass m is attached to a spring and a damper. The spring has a Hooke constant k and the damping is proportional to the speed through the damping constant b. The damping force acts in the direction opposite to the motion. Furthermore, the system is subject to an external force u. Let x denote the position of the mass. The continuous-time system dynamics is described by Newton's second law as:

$$m\ddot{x} = -kx - b\dot{x} + u \,, \tag{60}$$

which can equivalently be written as:

$$\ddot{x} + 2\zeta\omega\dot{x} + \omega^2 x = a \ , \tag{61}$$

where:

$$\omega = \sqrt{\frac{k}{m}} \tag{62}$$

$$\zeta = \frac{b}{2m\omega} \tag{63}$$

$$a = \frac{u}{m} \ . \tag{64}$$

The evolution of the position x of the mass is thus described by the position itself and the speed v as:

$$\left\{ \begin{array}{rcl} \dot{x} & = & v \\ \dot{v} & = & a - 2\zeta\omega v - \omega^2 x \ . \end{array} \right. \tag{65}$$

Optimization horizon. The optimization horizon T refers to the number of actions to be taken in the discrete process.

State space. The state is described at every time t by two variables: the position x_t and the speed v_t. The state space of the system is:

$$\mathcal{S} = \mathbb{R}^2 \ . \tag{66}$$

Initial state distribution. The initial states x_0 and v_0 are uniformly drawn from the intervals $[x_{0,min}, x_{0,max}]$ and $[v_{0,min}, v_{0,max}]$.

Action space. In its most general setting, the system can be submitted to any external acceleration a. However, we will only consider a discrete action space defined as follows:

$$\mathcal{A} = \{-0.3, -0.1, 0, 0.1, 0.3\} \ . \tag{67}$$

Disturbance space. We will consider a stochastic version of the problem where a real disturbance ξ_t is added to the action a_t such that an acceleration $a_t + \xi_t$ is applied to the system. In such a context, we have:

$$\xi_t \in \Xi = \mathbb{R} \ . \tag{68}$$

Disturbance distribution. The disturbance is sampled at time t from a Normal distribution centred at the current position x_t, and whose standard deviation is a linear combination of the magnitude of the action a_t and of the speed v_t:

$$P_\xi(\xi_t | s_t, a_t) = \mathcal{N}(\xi_t | x_t, 0.1 \times |a_t| + |s_t| + \epsilon) \ , \tag{69}$$

where ϵ is a constant equal to 10^{-6}.

Discrete dynamics. The discrete-time process comes from a discretization of the continuous process defined by Eqn. (65) with a discretization time-step $\Delta = 50$ms. The discrete dynamics f is the function computing the position and speed after a period Δ during which the constant acceleration $a_t + \xi_t$ is applied. The position x_{t+1} and the speed v_{t+1} can be computed from x_t and v_t using these analytical expressions:

$$x_{t+1} = g(x_t, v_t, a_t + \xi_t, \Delta), \tag{70}$$

$$v_{t+1} = \frac{\partial g}{\partial t}(x_t, v_t, a_t + \xi_t, t)|_{t=\Delta} \ , \tag{71}$$

where:

$$g(x_t, v_t, a, t) = \frac{a}{\omega^2} + \exp(-\zeta\omega t) \times$$
$$\left\{ \begin{array}{rl} (x_t - \frac{a}{\omega^2}) \cosh(\sqrt{\zeta^2 - 1}\omega t) + \frac{\frac{v_t}{\omega} + \zeta(x_t - \frac{a}{\omega^2})}{\sqrt{\zeta^2 - 1}} \sinh(\sqrt{\zeta^2 - 1}\omega t) & , \text{if} \quad \zeta > 1 \\ (x_t - \frac{a}{\omega^2}) + \big(v_t + \omega(x_t - \frac{a}{\omega^2})\big)t & , \text{if} \quad \zeta = 1 \\ (x_t - \frac{a}{\omega^2}) \cos(\sqrt{1 - \zeta^2}\omega t) + \frac{\frac{v_t}{\omega} + \zeta(x_t - \frac{a}{\omega^2})}{\sqrt{1 - \zeta^2}} \sin(\sqrt{1 - \zeta^2}\omega t) & , \text{if} \quad 0 < \zeta < 1 \ . \end{array} \right. \tag{72}$$

Reward function. The reward function is defined as:

$$\rho(a_t, s_t, \xi_t) = \exp\Big(-|x_t - x_{eq}| - (\omega - c_\omega)^2 - (\zeta - c_\zeta)^2 - \prod_{k=1}^{K} (\phi_k - c_k)^2 \Big) , \tag{73}$$

where ω, ζ and ϕ_k are parameters of the system that need to be optimized. Furthermore x_{eq}, c_ω, c_ζ, K and c_k are constant values. Let us also remark that the reward function does not depend on the disturbance.

The first term of the exponential will be minimized if the mass is stabilized at the position x_{eq}. The second and third terms are minimized if the parameters ω and ζ are equal to c_ω and c_ζ, respectively. The last term is a strictly positive function minimized if, at least one of the parameters ϕ_k equals the value c_k. Minimizing these terms results in maximizing the reward. Furthermore, since the the reward function is the exponential of a negative value, the reward is bounded by $r_{max} = 1$.

Parametrized MSD environment. A parametrized MSD environment is an environment $(\mathcal{S}, \mathcal{A}, \Xi, P_0, f_\psi, \rho_\psi, P_\xi, T)$ parametrized by the real vector $\psi = (\omega, \zeta, \phi_0, \phi_1, \phi_2) \in \mathbb{R}^5$.

Numerical values. In this work, we will consider the values given in Table 1 for the constant parameters.

Table 1: Parameters for the MSD.

Symbol	Value
$x_{0,min}$	0.198
$x_{0,max}$	0.202
$v_{0,min}$	−0.010
$v_{0,max}$	0.010
x_{eq}	0.200
c_ω	0.500
c_ζ	0.500
K	3.000
c_0	0.500
c_1	−0.300
c_2	0.200
T	100

D Optimal design of a solar off-grid microgrid

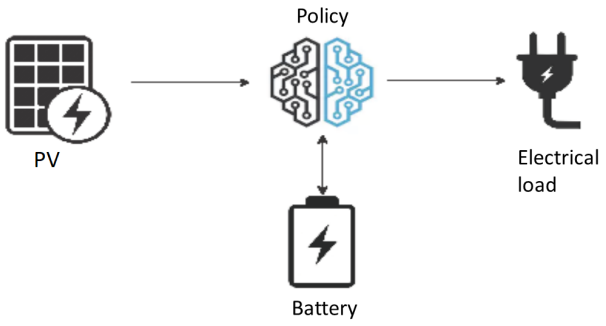


Figure 4: Microgrid configuration

A solar off-grid microgrid is a small-scale electrical grid composed of photovoltaic (PV) panels (converting solar energy into electricity) and a battery for ensuring the supply of an electrical load. A

schematic of the considered configuration is presented in Fig 4. The total cost of the microgrid is the sum of the investment costs and the penalties obtained for shedding the load if there is insufficient electricity available. In this section, we are interested in sizing the microgrid components, i.e. identifying the optimal investment in equipment that leads to the least total cost over the investment lifetime, assuming that the microgrid is operated in an optimal way.

This problem is therefore related to the one addressed in this paper, by noticing that finding the optimal investment (i.e., the size of the PV panels and the battery) is equivalent to optimizing both the "solar off-grid microgrid" environment and the policy at the same time. We note that the actions that can be taken by the policy are related to the charging/discharging power of the battery. An optimal policy should, in principle, charge the battery when there is an excess of solar power generated by the PV panels, and discharge that power from the battery when the electrical demand cannot be fully covered by the PV panels.

We will now provide, hereafter, a formalization of this problem that exactly fits the generic problem tackled in this paper. We note that more generic formalizations may exist, as for example those where the load consumption and the PV production cannot be considered as variables fully conditioned on the hour of the day, as will be assumed here. Those stand beyond the scope of this paper, even if they could lead to other interesting problem statements. Before carefully defining this benchmark problem, let us emphasize that we will use the notation $[\cdot]$ to indicate the corresponding unit of the symbol preceding it. In this section, $[W]$ denotes instantaneous power production in Watts, $[W_p]$ denotes nameplate (manufacturer) power capacity, $[Wh]$ denotes energy in Watt-hours and $[Wh_p]$ denotes nameplate (manufacturer) energy capacity. We now define the different elements of this learning optimal environment type of problem.

Optimization horizon. The optimization horizon is denoted by the value T.

State space. The state of the system can be fully described by $s_t = (SoC_t, h_t, \bar{P}_t^{C,h}, \bar{P}_t^{PV,h}) \in \mathcal{S} = \left[0, \overline{SoC}\right] \times \{0, ..., 23\} \times \mathbb{R}^+ \times \mathbb{R}^+$, where, at time t:

- $SoC_t\,[Wh] \in \left[0, \overline{SoC}\right]$ denotes the state of charge of the battery. The installed capacity of the battery is denoted by $\overline{SoC}\,[Wh_p] \in \mathbb{R}^+$.
- $h_t\,[h] \in \{0, ..., 23\}$ denotes the hour of the day.
- $\bar{P}_t^{C,h}\,[W] \in \mathbb{R}^+$ denotes the expected value of the electrical consumption level during hour h that is considered to be known.
- $\bar{P}_t^{PV,h}\,[W] \in \mathbb{R}^+$ denotes the expected value of the PV power generation during hour $h = h_t$ that is also considered to be known.

Initial state distribution. The initial state of charge SoC_0 is drawn uniformly from the interval $\left[0, \overline{SoC}\right]$ and the initial hour h_0 takes the value zero with probability one. The initial value for $\bar{P}_0^{C,h}$ is given by the first line of Table 3 in the corresponding column. Let $\overline{P^{PV}}\,[W_p] \in \mathbb{R}^+$ denote the capacity of PV panels installed, the column $\bar{p}^{PV,h}$ in Table 3 gives the average PV production per installed capacity (%). Subsequently, the initial value for $\bar{P}_0^{PV,h}$ is given by the product of $\overline{P^{PV}}$ and the first element of column $\bar{p}^{PV,h}$ in Table 3.

Action space. As previously described, the available actions correspond to defining the charging/discharging power of the storage system. The charging power is denoted by $P^B \in \left[-\overline{P^B}, \overline{P^B}\right]$, which will be positive during charging and negative during discharging. The charging/discharging limit $\overline{P^B} \in \mathbb{R}^+$ is assumed to be a proportion p (%) of the battery capacity as $\overline{P^B} = p \cdot \overline{SoC}$.

We therefore consider the continuous action space:

$$\mathcal{A} = \left[-\overline{P^B}, \overline{P^B}\right] . \tag{74}$$

Disturbance space. We consider as disturbance the variable $\xi_t = E_t^{C,h} \in \Xi \subseteq \mathbb{R}$, the stochastic deviation from the expected consumption for hour h_t.

Disturbance distribution. The disturbance is sampled at time t from a Normal distribution centred at zero with standard deviation $\sigma_{C,h}$ depending on the hour $h = h_t$:

$$P_\xi(\xi_t|s_t, a_t) = \mathcal{N}(\xi_t|0, \sigma_{C,h}) \ . \tag{75}$$

The values of the standard deviations $\sigma_{C,h}$ are given in Table 3 for every hour h of the day.

Transition function. We use a discretization time-step Δt of one hour for defining the discrete-time dynamics. For the state variable h we have therefore:

$$h_{t+1} = (h_t + 1) \mod 24 \ . \tag{76}$$

The state of charge of the battery is updated using a linear water tank model [Boukas et al., 2020]. With this tank model, the value of SoC_{t+1} at time $t+1$, if there were no limits on it, would be equal to A_{t+1} defined as follows:

$$A_{t+1} = SoC_t + \Delta t \cdot \begin{cases} \eta_{ch} \cdot P_t^B & \text{, if } P_t^B \geq 0 \\ P_t^B / \eta_{dis} & \text{, if } P_t^B < 0 \ , \end{cases} \tag{77}$$

where $\eta_{ch} \in [0, 1]$, $\eta_{dis} \in [0, 1]$ represent the charging and discharging efficiencies of the storage system. Given the fact that the state of charge of the battery lies within predefined limits, its state of charge at time $t+1$ is therefore defined as:

$$SoC_{t+1} = \begin{cases} 0 & \text{, if } A_{t+1} < 0 \\ \overline{SoC} & \text{, if } A_{t+1} \geq \overline{SoC} \\ A_{t+1} & \text{otherwise} \ . \end{cases} \tag{78}$$

The variable $\bar{P}_{t+1}^{C,h}$ takes the value reported in Table 3 at the line corresponding to the hour $h = h_{t+1}$. Finally, the variable $\bar{P}_{t+1}^{PV,h}$ is updated as:

$$\bar{P}_{t+1}^{PV,h} = \bar{p}^{PV,h} \cdot \overline{P^{PV}} \ , \tag{79}$$

where $\bar{p}^{PV,h}$ take the values reported in Table 3 at the line corresponding to the hour $h = h_{t+1}$.

Reward function. The reward signal is, in this case, a cost function composed of two parts, namely the investment cost and the operational cost. The reward signal is given by:

$$r_t = \rho(s_t, a_t, \xi_t) = -(c_t^{fix} + c_t^{shed}) \ , \tag{80}$$

where c_t^{fix} [\$] $\in \mathbb{R}^+$ represents a fixed hourly payment for settling the initial investment cost and c_t^{shed} [\$] $\in \mathbb{R}^+$ corresponds to the cost of shedding load at each time-step t.

In order to compute the fixed cost term c_t^{fix} we proceed as follows. Let c^{PV} [\$/W_p] $\in \mathbb{R}^+$ denote the cost per unit of PV capacity installed. The total installation cost for PV I^{PV} [\$] $\in \mathbb{R}^+$ is defined as:

$$I^{PV} = c^{PV} \cdot \overline{P^{PV}} \ . \tag{81}$$

Let c^B [\$/Wh_p] $\in \mathbb{R}^+$ denote the cost per unit of storage capacity installed. The total installation cost for battery storage I^B [\$] $\in \mathbb{R}^+$ is defined as:

$$I^B = c^B \cdot \overline{SoC} \ . \tag{82}$$

The investment cost I is the sum of the investment costs for each component of the microgrid defined as:

$$I = I^B + I^{PV} \ . \tag{83}$$

This payment occurs once in the beginning of the investment. In this case, we assume this investment to be a loan in its entirety. A fixed yearly payment P over the lifetime of the investment for settling the initial loan, is given by the following amortization formula:

$$P = I \frac{r(1+r)^n}{(1+r)^n - 1} \ , \tag{84}$$

where n is the number of years considered for the lifetime of the investment and $r(\%)$ is the interest rate considered. By noting that a common (non-leap) year has 8760 hours, we define the fixed hourly cost as:

$$c_t^{fix} = \frac{P}{8760} \,. \tag{85}$$

In order to compute the shedding cost term c_t^{shed} we proceed as follows. The realization of the consumption $P_t^{C,h}\,[W] \in \mathbb{R}^+$, after an action is taken at each time-step $t \in T$, corresponds to the actual consumption level in the interval $]t, t+1]$, i.e. for hour h_t. This variable takes the value:

$$P_t^{C,h} = \bar{P}_t^{C,h} + E_t^{C,h} \,, \tag{86}$$

where $h = h_t$ is the hour of the day at time t.

We denote by $\tilde{P}_t^B$ the actual charging power that can be applied to the battery considering its limited capacity. Given an action to charge P_t^B, the actual charge $\tilde{P}_t^B$ is constrained by the battery capacity limit for charging the available energy stored in the battery for discharging, according to:

$$\tilde{P}_t^B = \begin{cases} (\overline{SoC} - SoC_t)/\eta_{ch} & \text{, if } P_t^B > (\overline{SoC} - SoC_t)/\eta_{ch} \\ -(SoC_t) \cdot \eta_{dis} & \text{, if } P_t^B < -(SoC_t) \cdot \eta_{dis} \\ P_t^B & \text{otherwise .} \end{cases} \tag{87}$$

At each time-step t in the simulation horizon, there exists a power balance between the injections and the off-takes. The residual power resulting from the mismatch between production and consumption is curtailed $P_t^{curtail}\,[W] \in \mathbb{R}^+$. Formally the power balance is given by:

$$P_t^{curtail} = \bar{P}_t^{PV,h} - P_t^{C,h} - \tilde{P}_t^B \,. \tag{88}$$

If $P_t^{curtail}$ is positive, the excess of generation is simply lost (curtailed). If $P_t^{curtail}$ is negative, there is a lack of generation and a part of the load has to the shed. This is associated with a cost of shedding load $c_t^{shed}[\$] \in \mathbb{R}^+$ equal to:

$$c_t^{shed} = -\min(0, P_t^{curtail}) \cdot \pi^{shed} \,, \tag{89}$$

where $\pi^{shed}\,[\$/W] \in \mathbb{R}^+$ corresponds to the penalty per unit of power shed.

Parametrized environment. The off-grid microgrid environment $(\mathcal{S}, \mathcal{A}, \Xi, P_0, f_\psi, \rho_\psi, P_\xi, T)$ will be parametrized by the vector $\psi = (\overline{SoC}, \overline{P^{PV}}) \in \mathbb{R}^{+2}$.

Numerical values. Table 2 summarises the parameter values used in the experiments presented in this paper.

Table 2: Parameters for the solar off-grid microgrid.

Symbol	Value	Unit
η_{ch}, η_{dis}	75	%
σ^C, σ^{PV}	0.01	Wh
p	100	%
Δt	1	hour
c^{PV}	1	$\$/W_p$
c^B	1	$\$/W_p$
r	7	%
n	2	years
π^{shed}	10	$\$/Wh$
T	120	hour

Table 3: Electrical load consumption and PV production power factor data.

Hour	$\bar{P}^{C,h}$	$\sigma^2_{C,h}$	$\bar{p}^{PV,h}$
0	6.9	0.55	0.
1	6.4	0.50	0.
2	6.1	0.43	0.
3	5.9	0.39	0.
4	5.7	0.39	0.
5	5.4	0.37	0.
6	4.8	0.37	0.
7	4.5	0.36	0.
8	4.6	0.40	0.
9	4.6	0.43	0.04
10	4.7	0.44	0.08
11	4.9	0.47	0.12
12	5.1	0.42	0.14
13	5.3	0.40	0.15
14	5.4	0.42	0.14
15	5.4	0.47	0.12
16	5.4	0.43	0.08
17	5.8	0.44	0.04
18	8.4	0.81	0.
19	10.6	0.60	0.
20	11.0	0.55	0.
21	10.5	0.57	0.
22	9.2	0.60	0.
23	7.8	0.59	0.