Machine Learning
https://doi.org/10.1007/s10994-021-06020-8

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 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 number of benchmark strategies. Finally, the impact of the storage characteristics on the total revenues collected in the intraday market is evaluated.

Keywords European continuous intraday markets · Energy storage control · Markov decision process · Deep reinforcement learning · Asynchronous fitted Q iteration

Editors: Yuxi Li, Alborz Geramifard, Lihong Li, Csaba Szepesvari, Tao Wang.

✉ Ioannis Boukas
ioannis.boukas@uliege.be

[1] Department of Electrical Engineering and Computer Science, University of Liège, Liège, Belgium

[2] Global Market Analysis - GEM, ENGIE, Brussels, Belgium

Published online: 12 July 2021

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 and 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 and 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 and 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 and 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 and 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-min and 15-min products, and the gate closure takes place between five and 60 min 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 and 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 and 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-optimize the decisions made in the sequential trading floors from day-ahead to real time the problem has been traditionally addressed using multi-stage stochastic optimization. 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 optimization 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 and 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 optimization 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 and 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 and 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 and 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 and 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 and 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 optimized (Garnier and Madlener 2015). The optimization 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 and Hassler 2016). The ADP approach presented in Gönsch and 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 optimization.

The bidding strategy deployed by a storage device operator participating in a slightly different real-time market organized by NYISO is presented in Jiang and Powell (2014). In this market, the commitment decision is taken 1 h ahead of real-time and the settlements occur intra-hour every 5 min. 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 and Papavasiliou (2019). A threshold-based policy is used to optimize 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.

The *rolling intrinsic* benchmark was originally introduced in Gray and Khandelwal (2004) as a gas storage valuation method and relies on repeated re-optimization as new price information arrives. According to this method, the trader starts with an initial position and when new information about the prices arrives it calculates whether the profit of (partially) changing its position and taking the optimal position based on these new prices outweighs the transaction costs. The *rolling intrinsic* strategy yields profits if the spread between different tradable products changes sign and if it makes sense to swap trading decisions. This strategy, although risk-free, is not fundamentally maximizing profit.

1.3 Contributions of the paper

In this paper, we focus on the sequential decision-making problem related to the optimal operation of a storage device 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. Secondly, we model the asset trading process and describe explicitly the dynamics of the storage system.

We elaborate on a set of assumptions that allow the formulation of the resulting problem as an MDP. In particular, we consider that the strategy of the trading agents is modeled by a stochastic process that depends on the previous order book observations. The exogenous information to the trading process is considered to be the outcome of a time-dependent stochastic model and the charging/discharging decisions of the storage unit are always such that they minimize any resulting imbalances. Additionally, in order to reduce the possible trading actions, we assume that the trading agent can only select existing orders and is not able to post new free-standing offers (aggressor). In order to fully comply with German regulation policies, we further restrict the agent to select orders if and only if it does not result in any imbalances. Lastly, since in practice the storage unit is used for other operational obligations (reserves etc.), we consider that its initial and final state of charge for each day are decided in advance and are fixed during each CID trading session. This assumption allows the decoupling of the full optimization horizon in smaller (daily) windows.

Due to the high-dimensionality and the dynamically evolving size of the order book, we propose a novel low-dimensional order book representation that allows to capture the relevant order book information about the arbitrage opportunities of a storage unit. In particular, we pool the available orders and we engineer features that serve as a proxy of the potential benefit from this order book configuration for a storage device. Additionally, due to the dynamically evolving size of the order book, the set of possible actions is still large despite our assumption on our agent being an aggressor. We thus define a set of two high level actions, i.e. "Trade" and "Idle". The new action space allows us to design a set of policies that are variants of the *rolling intrinsic* strategy, where instead of sequentially repeating the optimization steps as new information arrives, we introduce the possibility to wait.

In the absence of a realistic model for the rest of the participants in the market we use historical data, to construct the trading environment in which the storage agent engages. The CID market 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). The resulting policy is evaluated using real data from the German CID market (EPEXSPOT 2017). The results suggest that the designed trading agent has the ability to identify the moments in which it would be better off by waiting based on a sequence of market indicators as well as other exogenous information. 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.
- We construct a novel state representation in order to provide a structured lower dimensional representation of the order book.

- We derive, using a batch-mode reinforcement algorithm, an operational policy that is able to identify the opportunity cost between trading and idling.

1.4 Outline of the paper

The rest of the paper is organized as follows. In Sect. 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 Sect. 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 Sect. 4. A case study using real data from the German CID market is performed in Sect. 5. The results as well as considerations about limitations of the developed methodology are discussed in Sect. 6. Finally, conclusions of this work are drawn and future recommendations are provided in Sect. 7. A detailed nomenclature is provided at the Appendix A.

2 Continuous intraday bidding process

In this section, we firstly present a detailed description of the CID market mechanism. Secondly, we model the dynamics and the decision-making process of an asset trading agent that participates in the CID market. The goal of the presented framework is to describe in a generic way the process under consideration. In the following sections, we introduce a number of assumptions and restrictions to this generic framework targeting a problem that can be tractable to solve.

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 $t_{close}(x)$. During the time interval $t \in \left[t_{open}(x), t_{close}(x)\right]$, 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 $s_t^{OB} = (o_j, \forall j \in N_t)$. 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 (hourly, half-hourly and quarter-hourly) $X = \{H_1, \ldots, H_{24}, HH_1, \ldots, HH_{48}, Q_1, \ldots, Q_{96}\}$, the corresponding continuous trading

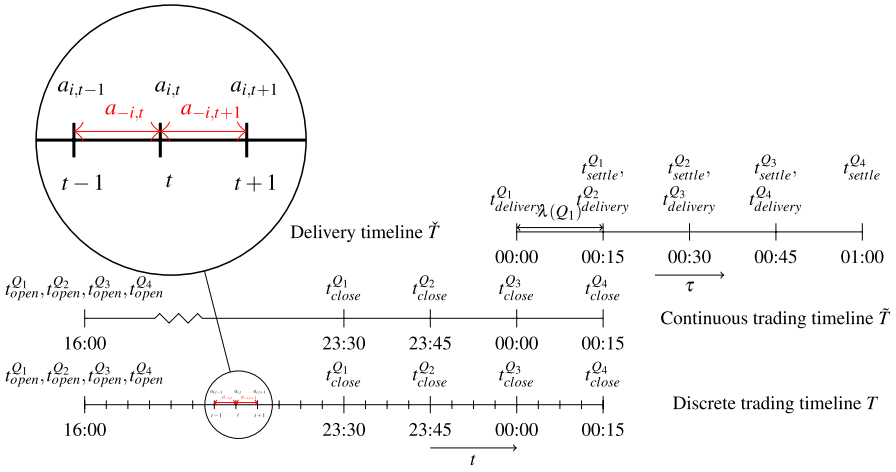

Fig. 1 Trading (continuous and discrete) and delivery timelines for products Q_1 to Q_4

horizon is defined as $\tilde{T} = \left[t_{open}(Q_1), t_{close}(Q_{96})\right]$. For instance, in the German CID market, trading of hourly (quarter-hourly) products for day D opens at 3 p.m. (4 p.m.) of day $D-1$ respectively. For each product x, the gate closes 30 min 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-min time slots from 00:00 until 01:00, is presented in Fig. 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 on the continuous trading horizon $\tilde{T}$. As shown in Fig. 1, the trading timeline $\tilde{T}$ is discretised in a high number of time-steps of constant duration Δt and results in the discretised timeline T. Each discretised trading interval for product x can be denoted by the set of time-steps $\hat{T}(x) = \left\{t_{open}(x), t_{open}(x) + \Delta t, \ldots, t_{close}(x) - \Delta t, t_{close}(x)\right\}$. Then, the discrete-time trading opportunities for the entire set of products X can be modelled such that the time-steps are defined as $t \in T = \bigcup_{x \in X} \hat{T}(x)$. 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:

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	

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

We define the state of the CID market environment at time-step t as $s_t^{OB} \in S^{OB}$. 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 $I = \{1, 2, \ldots, n\}$ 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 $a_t = (a_{1,t}, \ldots, a_{n,t}) \in \prod_{i=1}^{n} A_i$.

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 $y \in \{\}\}Sell'', \}\}Buy''\}$, the volume $v \in \mathbb{R}^+$, the price level $p \in \left[p_{min}, p_{max}\right]$ 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 execution specifications E includes:

- *All or Nothing* (AoN), where a limit order remains in the order book until it is entirely executed.
- *Fill or Kill* (FoK), where the entire volume of the limit order will be either matched immediately upon submission or withdrawn from the market.
- *Immediate or Cancel* (IoC), where as much as possible of the limit order volume is matched immediately upon submission and the remaining volume is withdrawn from the market.
- *Iceberg order* (IBO), that is a type of limit order, usually with a large volume, intended to hide the full size of the order by dividing it into a set of orders with smaller volumes. Each volume is shown to the market progressively, only after the previous one has been fully matched.
- *User-defined block order*, that is a limit order composed of one or several (up to 24) consecutive hourly products. User-defined block orders are AoN limit orders where only the entire volume may be executed.

The set of new orders is defined as $a_{i,t} = \left\{(x_j, y_j, v_j, p_j, e_j), \forall j \in N'_t \subseteq \mathbb{N}\right\}$. 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 $a_{-i,t} = (a_{1,t}, \ldots, a_{i-1,t}, a_{i+1,t}, \ldots, a_{n,t})$.

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 their submitted orders satisfy a number of conditions based on their limit prices as well as their execution specifications. For simple limit orders, a deal takes place 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. In the following, and for the sake of simplicity, we will assume that the agents can only submit simple limit orders ignoring the execution specifications.

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 $\{t_{close}(x), \ldots, t_{delivery}(x)\}$, 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 Eq. (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:

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

The discretisation of the delivery timeline $\check{T}$ is done with time-steps of duration $\Delta\tau$, equal to the minimum duration of delivery for the products considered. The discrete delivery timeline $\check{T}$ starts 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 quarter-hourly products are considered, as shown in Fig. 1, the delivery time-step is $\Delta\tau = 15$ min

and the delivery timeline $\check{T} = \{00:00, 00:15, \ldots, 01:00\}$, where $\tau_{init} = 00:00$ and $\tau_{term} = 01:00$. In general, when only one type of product is considered (e.g. quarter-hourly), 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 quarter-hourly 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 $\bar{T}(t) \subseteq \check{T}$ be a function that yields the subset of delivery time-steps $\tau \in \check{T}$ that follow time-step $t \in T$ such that:

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

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 $G_{i,t}(\tau) \in \left[\underline{G_i}, \overline{G_i}\right]$ 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 $C_{i,t}(\tau) \in \left[\underline{C_i}, \overline{C_i}\right]$, 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:

$$g_{i,t}(t') = G_{i,t}(\tau), \tag{4}$$

$$\begin{aligned} c_{i,t}(t') &= C_{i,t}(\tau), \\ \forall t' &\in [\tau, \tau + \Delta\tau). \end{aligned} \tag{5}$$

  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:

$$G_{i,t+1}(\tau) = G_{i,t}(\tau) + \Delta G_{i,t}(\tau), \tag{6}$$

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

Let $w_{i,t}^{exog}$ 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 $w_{i,t}^{exog}$. We define the residual production $P_{i,t}^{res}(\tau) \in \mathbb{R}$ at delivery time-step τ as the difference between the production and the consumption levels and can be computed by:

$$P_{i,t}^{res}(\tau) = G_{i,t}(\tau) - C_{i,t}(\tau). \tag{8}$$

We note that the amount of residual production $P_{i,t}^{res}(\tau)$ 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:

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

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 Eq. (9) written for time-step $t + 1$ is given by:

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

It can be observed that by substituting Eqs. (3), (6) and (7) in Eq. (10) we have:

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

The combination of Eqs. (8) and (9) with Eq. (11) yields the update of the imbalance vector according to:

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

- *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:

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

- *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 $\tau \in \breve{T}$ 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:

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

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 a function of the battery conditions. As can be observed from Eq. (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:

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

Combining Eqs. (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:

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

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 Eq. (12).

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

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

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 $u_{i,t} = (a_{i,t}, (\Delta C_{i,t}(\tau), \Delta G_{i,t}(\tau), \forall \tau \in \bar{T}(t))$.

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:

$$\begin{aligned} \left(\Delta C_{i,t}(\tau), \Delta G_{i,t}(\tau), \forall \tau \in \bar{T}(t)\right)^* = \arg\min \sum_{\tau \in \bar{T}(t)} | \Delta_{i,t+1}(\tau) |, \\ \text{s.t. (2), (3), (8), (9), (12), (14) .} \end{aligned} \tag{17}$$

One can easily see that from Eq. (11) this decision rule is equivalent to imposing $P_{i,t+1}^{res}(\tau)$ as close as possible to $P_{i,t+1}^{mar}(\tau)$, 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 $P_i^{res}(\tau)$ for delivery time-step τ is given by $P_i^{res}(\tau) = P_{i,t=t_{close}(x)}^{res}(\tau)$. 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 optimization 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:

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

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

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 $\rho : S^{OB} \times A_1 \times \cdots \times A_n \to \mathbb{R}$,
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 Fig. 1. We define the imbalance settlement timeline T^{Imb}, as $T^{Imb} = \left\{\tau + \Delta\tau, \forall\tau \in \check{T}\right\}$. 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:

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

2.5 Trading policy

All the relevant information that summarises the past and that can be used to optimize the market participation is assumed to be contained in the history vector $h_{i,t} = (s_{i,0}, a_{i,0}, r_{i,0}, \ldots, s_{i,t-1}, a_{i,t-1}, r_{i,t-1}, s_{i,t}) \in H_i$. 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 $\zeta = (s_{i,0}, a_{i,0}, \ldots, a_{i,K-1}, s_{i,K})$ 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:

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

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:

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

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 Markov decision process 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 is the only agent interacting with the CID market at each time-step t. Moreover, it is assumed that the other agents $-i$ can only react in the market according to the previously observed order book states. More precisely their actions $a_{-i,t}$ depend strictly on the history of order book states s^{OB}_{t-1} 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 Markov Game (Littman 1994). The process under consideration is such that it interleaves between agent i taking actions $a_{i,t}$ followed by its opponents $-i$ taking actions $a_{-i,t}$. Furthermore, the joint strategy of the opponents is modeled with Eq. (23) such that the agent i is involved in an MDP. This behaviour is illustrated in Fig. 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 $w^{exog}_{i,t}$ 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:

$$w^{exog}_{i,t} = b(w^{exog}_{i,t-1}, \ldots, w^{exog}_{i,t-k}, e_t), \tag{24}$$

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

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 Sect. 2.3.

As described in Sect. 2.3, the original control action that can be applied at each time-step t is $u_{i,t} = (a_{i,t}, (\Delta C_{i,t}(\tau), \Delta G_{i,t}(\tau), \forall \tau \in \check{T}))$. 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.

3.2 Decision process

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 optimization 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 optimization.
- *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 Eq. (23)
2. The state of the order book s^{OB}_{t+1} follows the transition given by Eq. (1)
3. The exogenous information $w^{exog}_{i,t}$ is given by Eq. (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 Eqs. (1), (6)–(12) (24), (25) and (16)
5. The instantaneous reward $r_{i,t}$ collected after each transition is given by Eqs. (18) and (19).

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

3.3 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^{red}_i be the space that contains only actions that match pre-existing orders in the order book. According to Assumption (4), the ith agent, at time-step t, is restricted to select actions $a_{i,t} \in A^{red}_i \subset A_i$. Let $s^{OB}_t = \left\{ (x^{OB}_j, y^{\prime OB}_j, v^{OB}_j, p^{OB}_j), \forall j \in N_t \right\}$ be the order book observation at trading time-step t. We use $y^{\prime OB}$ to denote that the new orders have the opposite side ("Buy" or "Sell") than the existing orders. We denote as $a^j_{i,t} \in [0, 1]$ the fraction of the volume accepted from order j. The reduced action space A^{red}_i is then defined as:

$$A^{red}_i = \{ \left(x^{OB}_j, y^{\prime OB}_j, a^j_{i,t} \cdot v^{OB}_j, p^{OB}_j \right), a^j_{i,t} \in [0, 1], \forall j \in N_t \}.$$

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

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

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^{OB}_t and the vector of fractions $\bar{a}_{i,t}$ and is given by:

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

3.4 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 and Hoffmann 2016), the imbalance market should not be considered as an optimization floor and the storage device should always be balanced at each trading time-step t ($\Delta_{i,t}(\tau) = 0, \forall \tau \in \check{T}$). 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 optimization of the storage unit over a long trading horizon can be decomposed into shorter optimization 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 Sect. 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 Sect. 4.1. Based on this dataset, we propose in Sect. 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 Sect. 4.3. Therefore, as part of the methodology, in Sect. 4.4 we propose a way for reducing the action space. Afterwards, in Sect. 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 optimized separately and thus, trading for one day corresponds to one trajectory. We consider that the trading horizon defined in Sect. 2.2 consists of K discrete trading time-steps such that $T = \{0, \dots, K\}$. A single trajectory sampled from the MDP described in Sect. 3 is defined as:

$$\zeta_m = \left(h^m_{i,0}, a^m_{i,0}, r^m_{i,0}, \dots, h^m_{i,K-1}, a^m_{i,K-1}, r^m_{i,K-1}, h^m_{i,K}\right).$$

A set of M trajectories can be then defined as:

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

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

$$F' = \begin{Bmatrix} (h^1_{i,0}, a^1_{i,0}, r^1_{i,0}, h^1_{i,1}), & \cdots & (h^1_{i,K-1}, a^1_{i,K-1}, r^1_{i,K-1}, h^1_{i,K}), \\ \vdots & \ddots & \vdots \\ (h^M_{i,0}, a^M_{i,0}, r^M_{i,0}, h^M_{i,1}), & \cdots & (h^M_{i,K-1}, a^M_{i,K-1}, r^M_{i,K-1}, h^M_{i,K}) \end{Bmatrix}.$$

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

$$F'_t = \left\{(h^m_{i,t}, a^m_{i,t}, r^m_{i,t}, h^m_{i,t+1}), m = 1, \dots, M\right\}_t, \quad \forall t \in \{0, \dots, K-1\}.$$

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 optimization of the MDP defined in Sect. 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:

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

A time-variant policy $\pi = \left\{\mu_0, \dots, \mu_{K-1}\right\} \in \Pi$, consists in a sequence of functions μ_t, where $\mu_t : H_i \to A_i^{red}$. 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 $\pi^{t+1} = \left\{\mu_{t+1}, \dots, \mu_{K-1}\right\}$ 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 Eq. (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:

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

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}:

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

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

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

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 Eq. (30). However, two aspects render the use of the standard value iteration algorithm impossible for solving the MDP defined in Sect. 3. First, the transition probabilities of the MDP defined in Sect. 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 and 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 Sect. 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 $F_t' = \left\{ (h_{i,t}^m, a_{i,t}^m, r_t^m, h_{i,t+1}^m), m = 1, \ldots, M \right\}$ obtained from previous experience is exploited in order to update the parameter vectors θ_t by solving the supervised learning problem presented in Eq. (31). 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 Eq. (32). 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 $\hat{Q} = \{\hat{Q}_0, \ldots, \hat{Q}_{K-1}\}$ until termination at $t = 0$.

$$\theta_t^* = \arg\min_{\theta_t} \sum_{m=1}^{M} (Q_t(h_{i,t}^m, a_t^m; \theta_t) - y_t^m)^2 \tag{31}$$

$$y_t^m = r_t^m + \max_{a_{i,t+1} \in A_i^{red}} Q_{t+1}(h_{i,t+1}^m, a_{i,t+1}; \theta_{t+1}^*) \tag{32}$$

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

$$\begin{aligned} \hat{\mu}_t^*(h_{i,t}) = & \arg\max_{a_{i,t} \in A_i^{red}} Q_t(h_{i,t}, a_{i,t}; \theta_t^*), \\ & \forall t \in \{0, \ldots, K-1\}. \end{aligned} \tag{33}$$

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

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

is composed of :

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

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

The state space H_i as well as the action space A_i^{red}, as described in Sect. 3.3, 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 $p : A_i' \rightarrow A_i^{red}$, from space A_i' to the reduced action space A_i^{red}. The high-level actions are defined as follows:

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 Eq. (19), consists only of the trading revenues obtained from the matching process of orders at time-step t. We will further assume that mapping $u : \mathbb{R}^+ \times \{\text{“}Sell\text{”}, \text{“}Buy\text{”}\} \rightarrow \mathbb{R}$ 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:

$$u(v^{OB}, y^{OB}) = \begin{cases} v^{OB}, & \text{if } y^{OB} = \text{“}Buy\text{”}, \\ -v^{OB}, & \text{if } y^{OB} = \text{“}Sell\text{”}. \end{cases} \tag{34}$$

Consequently, the reward function ρ defined in Sect. 2.4 is adapted according to the proposed modifications. The new reward function ρ, where $\rho : S^{OB} \times \bar{A}_i^{red} \rightarrow \mathbb{R}$, 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:

$$r_{i,t} = \rho\left(s_t^{OB}, \bar{a}_{i,t}\right) = \sum_{j=1}^{N_t} a_{i,t}^j \cdot u(v_j^{OB}, y_j^{OB}) \cdot p_j^{OB}. \tag{35}$$

The High-level action “Trade” amounts to solving the bid acceptance optimization problem presented in Model 1. The objective function of the problem, formulated in Eq. (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 $\hat{N}_t(\tau) \subset \mathbb{N}$ the set of unique indices of the orders available at step t that correspond to delivery time-step τ and $N_t = \bigcup_{\tau \in \bar{T}} \hat{N}_t(\tau)$. In Eq. (38), the energy purchased and sold $(\sum_{j \in \hat{N}_t(\tau)} a_{i,t}^j u(v_j^{OB}))$, 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 Eq. (38), 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 Eqs. (40) to (44). The binary variables $k_{i,t} = (k_{i,t}(\tau), \forall \tau \in \check{T})$ 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 optimal vector of fractions:

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

that are used in Eq. (26) to construct the action $a_{i,t} \in A^{red}_i$. 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.

Model 1 "Trade"

Input: t, s^{OB}_t, $P^{mar}_{i,t}$, $\underline{SoC_i}$, $\overline{SoC_i}$, $\underline{C_i}$, $\overline{C_i}$, $\underline{G_i}$, $\overline{G_i}$, η, SoC^{init}_i, SoC^{term}_i, 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:

$$\max_{\substack{\bar{a}_{i,t},\, SoC_{i,t+1} \\ G_{i,t+1},\, C_{i,t+1} \\ \Delta G_{i,t},\, \Delta C_{i,t} \\ k_{i,t+1},\, r_{i,t}}} \sum_{j \in N_t} a^j_{i,t} \cdot u(v^{OB}_j, y^{OB}_j) \cdot p^{OB}_j \tag{37}$$

$$\text{s.t.} \sum_{j \in \bar{N}_t(\tau)} a^j_{i,t} u(v^{OB}_j, y^{OB}_j) + P^{mar}_{i,t}(\tau) +$$

$$C_{i,t+1}(\tau) = G_{i,t+1}(\tau), \qquad \forall \tau \in \bar{T}(t) \tag{38}$$

$$SoC_{i,t+1}(\tau + \Delta\tau) = SoC_{i,t+1}(\tau) + \Delta\tau \cdot \left(\eta \cdot C_{i,t+1}(\tau) - \frac{G_{i,t+1}(\tau)}{\eta}\right), \quad \forall \tau \in \bar{T}(t) \tag{39}$$

$$\underline{SoC_i} \le SoC_{i,t+1}(\tau) \le \overline{SoC_i}, \qquad \forall \tau \in \bar{T}(t) \tag{40}$$

$$SoC^{init}_i = SoC_{i,t+1}(\tau_{init}), \tag{41}$$

$$SoC^{term}_i = SoC_{i,t+1}(\tau_{term}), \tag{42}$$

$$\underline{C_i} \le C_{i,t+1}(\tau) \le k_{i,t+1}(\tau) \cdot \overline{C_i}, \qquad \forall \tau \in \bar{T}(t) \tag{43}$$

$$\underline{G_i} \le G_{i,t+1}(\tau) \le (1 - k_{i,t+1}(\tau))\,\overline{G_i}, \quad \forall \tau \in \bar{T}(t) \tag{44}$$

$$G_{i,t+1}(\tau) = G_{i,t}(\tau) + \Delta G_{i,t}(\tau), \qquad \forall \tau \in \bar{T}(t) \tag{45}$$

$$C_{i,t+1}(\tau) = C_{i,t}(\tau) + \Delta C_{i,t}(\tau), \qquad \forall \tau \in \bar{T}(t) \tag{46}$$

$$k_{i,t+1}(\tau) \in \{0, 1\}, \qquad \forall \tau \in \bar{T}(t) \tag{47}$$

$$a^j_{i,t} \in [0, 1], \qquad \forall j \in N_t \tag{48}$$

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:

$$G_{i,t+1}(\tau) = G_{i,t}(\tau), \forall \tau \in \bar{T}(t), \tag{49}$$

$$C_{i,t+1}(\tau) = C_{i,t}(\tau), \forall \tau \in \bar{T}(t), \tag{50}$$

$$SoC_{i,t+1}(\tau) = SoC_{i,t}(\tau), \forall \tau \in \bar{T}(t). \tag{51}$$

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 when the agent learns to idle, given a current situation, it does not necessarily mean, that if it had chosen to "*Trade*" instead, it 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. In this way, we extend the myopic policy, which we refer to as the *rolling intrinsic* policy. According to the latter, the agent selects at every time-step t of the trading horizon the combination of orders that optimizes its operation and profits, based on the current information assuming that the storage device must remain balanced for every delivery period as presented in Gray and Khandelwal (2004), Löhndorf and Wozabal (2020) and Bertrand and Papavasiliou (2019). The *rolling intrinsic* policy is, thus, equivalent to sequentially selecting the action "*Trade*" (Model 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 Sect. 4.3, contains the entire history of all the relevant information available for the decision-making process up to time t. As such, the information contained in the trajectories is represented as unstructured sets. 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 an alternative form. This alternative form is engineered with the aim to capture the structure between observed bids in the order book.

First, the vector containing the entire history of actions is reduced to a vector of binary variables after the modifications introduced in Sect. 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 $s_t^{OB} = ((x_j^{OB}, y_j'^{OB}, v_j^{OB}, p_j^{OB}), \forall j \in N_t \subseteq \mathbb{N}) \in S^{OB}$ that is defined in Sects. 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 $\mid N_t \mid$, 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 Fig. 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 Fig. 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 Fig. 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 Fig. 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 Fig. 3a. As illustrated in Fig. 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 Figs. 2b and 3b.

For instance, we define as $D1$ the signed distance between 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 $s_t'^{OB} \in S'^{OB} = \{D1, \dots, D10\}$ 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 Sect. 4.3, $s_{i,t}^{private}$ is defined as:

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

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

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

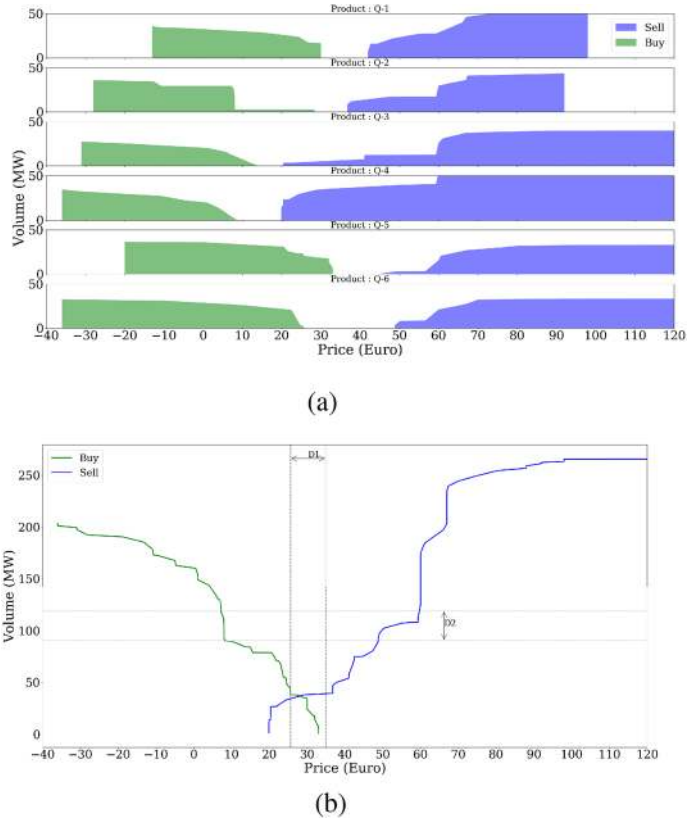

Fig. 2 **a** Market depth per product (for products Q_1 to Q_6) at a time-step t with arbitrage potential. **b** The corresponding aggregated curves for a profitable order book

always be zero according to the way the high-level actions are defined in Sect. 4.4. Additionally, Assumption (3) regarding the default strategy for storage control in combination with Assumption (5) yields a direct correlation between vectors $P^{mar}_{i,t}$ and $G_{i,t}$, $C_{i,t}$, $SoC_{i,t}$. Thus, it is considered that $P^{mar}_{i,t}$ 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 $z_{i,t} = (s'_{i,0}, a'_{i,0}, r_{i,0}, \ldots, a'_{i,t-1}, r_{i,t-1}, s'_{i,t}) \in Z_i$, where $s'_{i,t} = (s'^{OB}_t, P^{mar}_{i,t}, w^{exog}_{i,t}) \in S'_i$. This pseudo-state can be seen as the result of applying an encoder $enc : H_i \rightarrow Z_i$ which maps a true state $h_{i,t}$ to pseudo-state $z_{i,t}$.

$z_{i,t} = (s'_{i,0}, a'_{i,0}, r_{i,0}, \ldots, a'_{i,t-1}, r_{i,t-1}, s'_{i,t}) \in Z_i$

In the following, it is considered that the pseudo-state $z_{i,t} \in Z_i$ contains all the relevant information for the optimization of the CID market trading of an asset-optimizing agent. Thus, replacing the true state $h_{i,t}$ with pseudo-state $z_{i,t}$ is not considered to lead to a sub-optimal policy. The resulting decision process after the state and action spaces reductions is illustrated in Fig. 4.

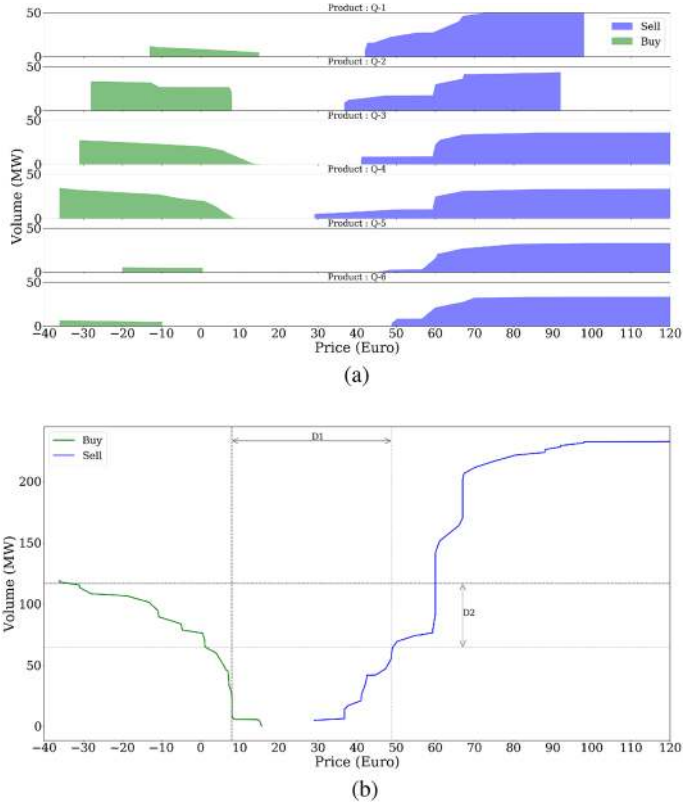

Fig. 3 **a** Market depth per product (for products Q_1 to Q_6) at a time-step t with no arbitrage potential. **b** The corresponding aggregated curves for a non profitable order book

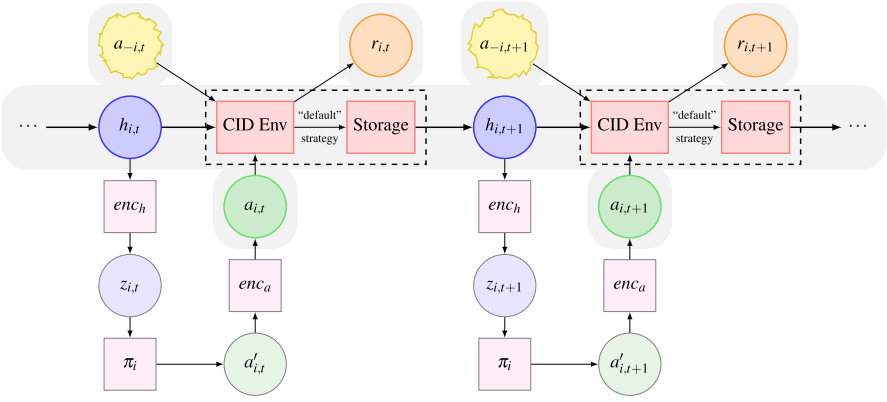

Fig. 4 Schematic of the decision process. The original MDP is highlighted in a gray background. The state of the original MDP $h_{i,t}$ is encoded in pseudo-state $z_{i,t}$. Based on $z_{i,t}$, agent i takes an high-level action $a'_{i,t}$, according to its policy π_i. This action $a'_{i,t}$ is mapped 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 $h_{i,t}$

4.6 Generation of artificial trajectories

Algorithm 1 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 $\hat{Q}, F$

In this section, the generation of artificial trajectories for addressing exploration issues in an offline setting is discussed. Indeed, if we were to implement an agent that selects at every time-step among the "Idle" and "Trade" actions, we would collect a certain number of trajectories (one per day) over a certain period of interactions with the real market. The collected dataset could be used to train a policy using a batch mode RL algorithm, as described in Sect. 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.

As discussed in Sect. 4.2, sufficient exploration of the state and action spaces is a key requirement for converging to a near-optimal policy. 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.

Furthermore, the aforementioned approach requires the direct interaction with the CID market in order to collect samples from the unknown initial state distribution and from the opponents' actions. In the RL context, exploration is then 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 7 (*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 (7) 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 (7) allows one to tackle the exploration issues discussed previously in an offline setting by generating several artificial trajectories using historical order

book data. An artificial trajectory is generated as follows. At each time t, the agent i takes an action according to the current state of the order book. Under Assumption (7), the next state of the order book is then the historical state at time $t+1$ from which the bids accepted by agent i have been removed. Finally, in this framework, such an artificial trajectory corresponds to a trajectory sampled from the CID model developed. We denote by $\hat{E}$ 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 $M = \hat{E} \cdot |L^{train}|$.

The simulation of trajectories is performed according to the process described in Figure 1 in Ernst et al. (2005). Nevertheless, in this framework, trajectories are generated artificially as described previously rather than directly sampled from the system. This process interleaves the generation of trajectories with the computation of an approximate Q-function using the trajectories already generated. As shown in Algorithm 1, for a number of episodes *ep*, we randomly select days from the train 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 $(1-\epsilon)$. 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 $\hat{E}$.

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

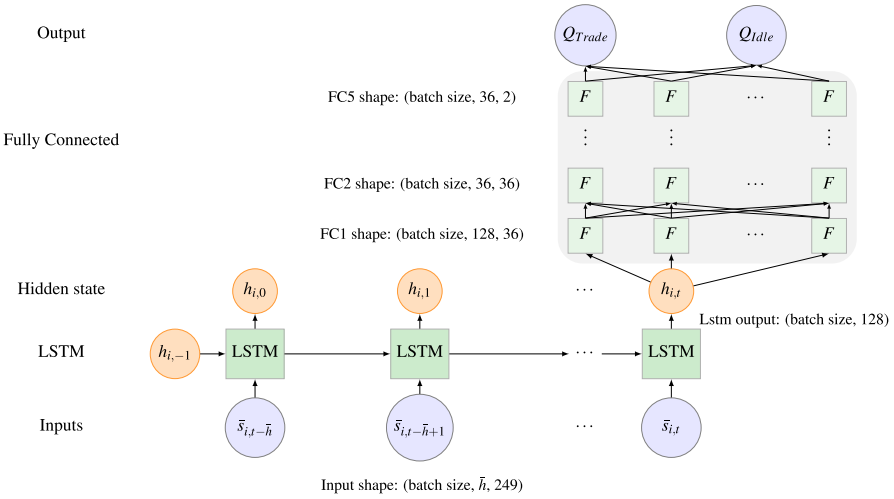

Fig. 5 Schematic of the neural network architecture

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 Fig. 5, each sample in the batch contains a vector of size 249 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 $z_{i,t} = (a'_{i,t-\bar{h}-1}, r_{i,t-\bar{h}-1}, s'_{i,t-\bar{h}}, a'_{i,t-\bar{h}}, r_{i,t-\bar{h}}, \ldots, a'_{i,t-1}, r_{i,t-1}, s'_{i,t}) \in Z_i$. 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 $z_{i,t} = (\bar{s}_{t-\bar{h}}, \ldots, \bar{s}_t)$.

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 optimization model is constructed and solved.

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

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

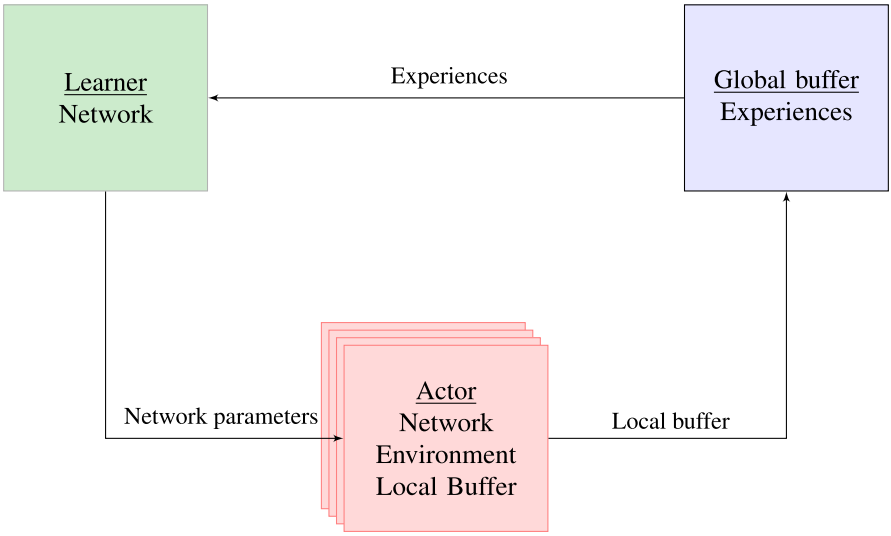

Fig. 6 Schematic of the asynchronous distributed architecture. Each actor runs on a different thread and contains a copy of the environment, an individual ϵ-greedy policy based on the latest version of the network parameters and a local buffer. The actors generate trajectories that are stored in their 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

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 to the case of a PHES unit. Firstly, the parameters and the exogenous information used for the optimization of the CID market participation of a PHES operator are described. Secondly, 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.

Table 3 Dataset columns

Symbol	Description
Instrument type	Product type i.e. quarter-hourly, half-hourly or hourly
Delivery instrument	Time of delivery (e.g. 00:00–00:15)
Delivery date	Date stamp (e.g. 17/04/2015)
Start validity date	Date-time stamp (e.g. 16/04/2016 17:58:47.093)
End validity date	Date-time stamp (e.g. 16/04/2015 23:30:02.880)
Cancelling date	Date-time stamp (e.g. 04/16/2015 23:30:02.880)
Is executed	Integer, i.e. 0 (not executed), 1 (fully executed) and 2 (partially executed)
Side	"Buy" or "Sell"
Price (€/MWh)	Limit price
Execution Price ($/MWh)	Price at which an order was matched
Volume (MWh)	Offered volume
Executed Volume (MWh)	Volume that was transacted
Parent ID	Unique order ID (10 digit number)
Initial ID	Unique order ID that was assigned at first appearance of the order

5.1 CID order book data

In the presented case study, we used a dataset containing anonymous historical orders for the quarter-hourly products of the German CID market for the second half of the year 2015. In particular, for the construction of the training/test sets, we proceed as following. Due to the high computational burden, we train our algorithm on a period of $|L^{train}| = 92$ days (i.e. 2015/05/01–2015/07/31) in which a high variance in prices is observed and therefore high profit potential. Subsequently, we evaluate its performance in the following $|L^{test}| = 152$ days (i.e. 2015/08/01–2015/12/31). This dataset is composed of the columns presented in Table 3. In this paper, we did not consider the execution specifications of limit orders. This choice was partially made (besides its increased modeling complexity), due to the fact that these specifications are not stated explicitly in the historical order book dataset that is used. In effect, it could be possible to apply some heuristic rules in order to identify some of the execution specifications ex-post. However, this task is not trivial, especially for some of the most complex specifications. Therefore, we chose to treat all orders as simple limit orders.

We simulate through the historical order book in such a way that at each trading step t, the agent is presented with the active orders, i.e. the historical orders for which trading step t is after their *Start Validity Date* and t is before their *End Validity Date*. Additionally, we filter the orders at each step in order to guarantee that for each product the minimum selling price (ask) is always higher that the maximum buying price (bid). The number of orders dropped by this filtering process depends on the trading frequency that is selected. In this case study, the dropped orders correspond to less than 1% of the orders observed during the defined trading timeline for the train and test sets. Table 4 , summarizes the information regarding the order book data that were used in the presented case study.

Table 4 Summary statistics of the order book dataset used in the case study

Δt	Total orders	Observed orders	Dropped
15 min	9,218,857	3,450,838	7904
5 min	9,218,857	4,226,184	13,074

5.2 Parameters specification

The proposed methodology is applied for an instance of a PHES unit[2] participating in the German CID market with the following characteristics:

- $\overline{SoC_i} = 40$ MWh,
- $\underline{SoC_i} = 0$ MWh,
- $\overline{SoC_i^{init}} = SoC_i^{term} = \left(\overline{SoC_i} - \underline{SoC_i}\right)/2$,
- $\overline{C_i} = \overline{G_i} = 8$ MW,
- $\underline{C_i} = \underline{G_i} = 0$ MW,
- $\overline{\eta} = 90\%$.

The discrete trading horizon has been selected to be the full day, i.e. $T = \{16:00, \ldots, 00:00, \ldots, 23:15\}$. The trading time interval is selected to be $\Delta t = 15$ min. Thus the trading process takes $K = 124$ steps until termination. Moreover, all 96 quarter-hourly products of the day, $X = \{Q_1, \ldots, Q_{96}\}$, are considered. Consequently, the delivery timeline is $\check{T} = \{00:00, \ldots, 23:45\}$, 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 min before the physical delivery of electricity begins (e.g. $t_{close}(Q_1) = 23:30$ etc.).

The total number of simulated episodes was selected to be $\hat{E} = 10000$ episodes for the artificial trajectories generation process, described in Sect. 4.6. During the trajectories generation process the high-level actions ("Trade" or "Idle") were chosen following an ϵ -greedy policy. As described in Sect. 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 $z_{i,t} = (s'_{i,0}, a'_{i,0}, r_{i,0}, \ldots, a'_{i,t-1}, r_{i,t-1}, s'_{i,t}) \in Z_i$ is composed of the entire history of observations and actions up to time-step t, as described in Sect. 4.5. For the sake of memory requirements, as explained in Sect. 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 $\bar{h}_{max} = 10$. Simulations were performed in an AWS EC2 g4dn.8xlarge instance with the following specifications: 32 cores Intel(R) Xeon(R) Platinum 8259CL CPU 2.50 GHz, 1 GPU NVIDIA T4, 128 Gb RAM. The operating system used was Ubuntu 18.04 and the programming language was Python 3.7. The optimization model is implemented on docplex, the CPLEX modeling framework for Python, and solved with CPLEX 12.10. Full training of one experiment takes approximately 6 h. For the full experiment training time is approximately 60 h.

[2] A small instance of the storage unit was selected due to the low volumes available in the historical order book data used.

5.3 Exogenous variable

The exogenous variable $w_{i,t}^{exog}$ represents any relevant information available to agent i about the system. In this case study, we assumed that the variable $w_{i,t}^{exog}$ 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
- The 96 values of the intraday auction prices for the entire trading day
- Time features: (1) the month, (2) whether the traded day is a weekday or weekend and (3) the time-step t.

5.4 Benchmark strategies

In this section, we introduce several benchmark strategies used to compare the performance of the fitted Q policy. Firstly, we consider a look-ahead policy that has access to future states of the MDP. Although this policy cannot be implemented in practice, it can provide a good upper bound on the performance of other non-anticipative strategies. Secondly, we introduce a heuristic policy that relies on the solution of the look-ahead policy on the train set in order to obtain a probabilistic decision rule. In addition to that, we use variants of the *rolling intrinsic* that selects the "Trade" action at each step. Finally, an alternative reinforcement learning algorithm is considered in order to show the benefits of using batch mode learning

5.4.1 Look-ahead

Firstly, we select an anticipative strategy as a benchmark which we call look-ahead policy π^{LA}. This policy cannot be implemented in practice because it relies on future information that a storage operator would not have in its possession in real-time. However, it can provide a good measure on how well the fitted Q iteration policy can anticipate future rewards.

According to the look-ahead policy π^{LA}, at each decision step t the agent can fast-forward a number of ψ steps into the future[3] and compute the potential profits $\hat{r}_{i,t+\psi}$. The computation of the potential profits $\hat{r}_{i,t}$ at each time-step t is performed by solving the optimization problem defined in Model 1 without applying any of the output actions to the real system. If the future potential profits $\hat{r}_{i,t+\psi}$ are higher than the current potential profits $\hat{r}_{i,t}$, the agent selects to "Idle". In the opposite case, the agent selects to "Trade". In the presented case study, we use the values $\psi = 1$ and $\psi = 2$, denoted as π^{LA_1} and π^{LA_2} respectively. The look-ahead policy π^{LA} can be summarized by the following rule:

$$a'_{i,t} = \pi^{LA}(\hat{r}_{i,t+\psi}, \hat{r}_{i,t}) = \begin{cases} \text{“}Idle\text{"}, & \text{if } \hat{r}_{i,t+\psi} \geq \hat{r}_{i,t}, \\ \text{“ }Trade\text{"}, & \text{otherwise .} \end{cases} \tag{52}$$

[3] Only changes related to the state of the order book occur during these steps.

5.4.2 Heuristic

Secondly, we consider a heuristic policy π^{HE} that relies on the solution obtained by the look-ahead policy on the train set in order to obtain a probabilistic decision rule. More specifically, for each time step t of the trading horizon T, we compute the percentage of days d of the train set L^{train} for which the look-ahead policy has selected the action ($a_{d,t}^{LA}$) to "Trade" as following:

$$p_t^{\text{"Trade"}} = \frac{\sum_{d \in L^{train}} \mathbb{1}_{\left\{a_{d,t}^{LA} = \text{"Trade"}\right\}}}{|L^{train}|} \tag{53}$$

We can construct a stochastic policy where at each time-step t, an action is sampled according to a Bernoulli distribution over the action space as following:

$$a'_{i,t} \sim \pi^{HE}(\cdot|t) = \begin{cases} \text{"}Trade\text{"}, & \text{with probability } p_t^{\text{"}Trade\text{"}}, \\ \text{"}Idle\text{"}, & \text{with probability } 1 - p_t^{\text{"}Trade\text{"}}. \end{cases} \tag{54}$$

5.4.3 Rolling intrinsic

Central in our comparison is the *rolling intrinsic* policy (Gray and Khandelwal 2004; Löhndorf and Wozabal 2020; Bertrand and Papavasiliou 2019), denoted by π^{RI}. This is a myopic policy according to which, the agent selects at each trading time-step t the action "Trade", as described in Sect. 4.4. Formally we have:

$$a'_{i,t} = \pi^{RI}(\cdot) = \text{"}Trade\text{"} \tag{55}$$

In addition to that, and for comparison purposes, we also consider a variant of the rolling intrinsic $\pi^{RI_{5min}}$, where the trading agent selects the action "Trade" at higher frequency ($\Delta t = 5$ min).

Selecting the action to "Trade" at each time-step t can be quite restrictive, especially due to the fact that in the beginning of the trading horizon there is not much trading activity taking place. However, later in the trading horizon and especially around the gate closures an upsurge in trading activity is expected as market participants attempt to finalize their positions. Motivated by this, we also consider a variant of the *rolling intrinsic* policy, denoted by π^{RI_g} that selects the action "Trade" only when time-step t corresponds to a gate closure for a product. More precisely, we have:

$$a'_{i,t} = \pi^{RI_g}(\cdot|t) = \begin{cases} \text{"}Trade\text{"}, & \text{if } t \in \left\{t_{close}(x), \forall x \in X_t\right\}, \\ \text{"}Idle\text{"}, & \text{otherwise.} \end{cases} \tag{56}$$

5.4.4 Learning-based

Finally, we consider an alternative reinforcement learning algorithm for comparison purposes. We select the asynchronous prioritized experience replay deep Q networks (APE-X DQN) algorithm, as it is presented in Horgan et al. (2018). Let $Q(z_t, a_t; \theta)$ denote a parametric approximation of the state-action value function Q with parameters θ. This algorithm proceeds by iteratively updating parameters θ. In each iteration step, the algorithm is using

the latest version of parameters θ_{old} and the one-step system transitions from set F to construct the target vector y according to:

$$y = r_t + \max_{a'_{i,t+1} \in A'_i} Q(z_{i,t+1}, a'_{i,t+1}; \theta^{old}) \tag{57}$$

Parameters θ are then updated based on the target vector according to:

$$\theta_{new} = \theta_{old} - \alpha \cdot (y - Q(z_t, a'_t; \theta_{old})) \cdot \nabla_\theta Q(z_t, a'_t; \theta), \tag{58}$$

This iterative process continues until the algorithm converges to the optimal parameters θ^*. Once the parameters θ^* are computed, the policy π^{APEXDQ} is obtained as:

$$\pi^{APEXDQN}(z_{i,t}) = \arg\max_{a'_{i,t} \in A'_i} Q(z_{i,t}, a'_{i,t}; \theta^*), \tag{59}$$

The main difference between the fitted Q algorithm proposed in this paper and APE-X DQN is that a single neural network is used for approximating the value function $Q(z_t, a'_t; \theta)$ instead of using one network per time-step t. The selection of this asynchronous algorithm is motivated by the computational efficiency that was discussed previously in Sect. 4.8. The neural network architecture used for approximating the value function is the same as the one described in Sect. 4.7.

Additionally, the benchmark presented in Bertrand and Papavasiliou (2019) could be used for comparison purposes. However, the basis of our analysis is significantly different. In particular, the assumptions related to the storage operation (Assumptions 5 and 6) as well as the fact that quarterly products are considered (instead of hourly) in this paper, constitute the comparison impossible.

5.5 Validation process

The performance of the policy obtained using the fitted Q iteration algorithm, denoted by π^{FQ}, is evaluated on test set L^{test} that contains historical data from 152 days. These days are not used during the training process. This evaluation pass is performed by using a greedy policy based on the outputs of the neural networks. During this process, the policy is fixed and is not updated with data collected from the test set. 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 (7) 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 benchmark policies described above. The comparison is based on the computation of the return of the policies on each day. For a given policy, the return over a day is simply computed by running the policy on the day and summing up the rewards obtained.

Our learning algorithm 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. Hence, we 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 a dataset, we will actually report the average performances of ten learned policies over this dataset.

We describe the different indicators that will be used afterwards to assess the performance of our method. These indicators are computed for both the train 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 train set.

Let $V_d^{\pi^b}$ denote the return of a policy $\pi^b \in \mathbb{B} = \left\{\pi^{FQ}, \pi^{APEXDQN}, \pi^{HE}, \pi^{LA_1}, \pi^{LA_2}, \pi^{RI_g}, \pi^{RI_{5min}}, \pi^{RI}\right\}$ for day d, respectively. Set $\mathbb{B}$ contains the policies under comparison in this case study. 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 $V_d^{\pi^b}$ of each policy $\pi^b \in \mathbb{B}$ 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 can compute the sum of returns over the entire set of days as follows:

$$V^{\pi^b} = \sum_{d \in L^{test}} V_d^{\pi^b}, \quad \forall \pi^b \in \mathbb{B} \tag{60}$$

Central in our analysis is the risk-averse *rolling intrinsic* policy. Therefore, we consider an alternative performance indicator defined as the discrepancy of the returns coming from the considered policies with respect to the *rolling intrinsic*. We define the profitability ratio $r_d^{\pi^b}$ for each day $d \in L^{test}$, that corresponds to the signed percentage difference between the two policies as follows:

$$r_d^{\pi^b} = \frac{V_d^{\pi^b} - V_d^{\pi^{RI}}}{V_d^{\pi^{RI}}} \cdot 100\%, \quad \forall \pi^b \in \mathbb{B}. \tag{61}$$

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 two policies, as:

$$r_{sum}^{\pi^b} = \frac{V^{\pi^b} - V^{\pi^{RI}}}{V^{\pi^{RI}}} \cdot 100\%, \quad \forall \pi^b \in \mathbb{B}. \tag{62}$$

5.6 Results

The performance indicators described previously are computed for both the training and the test set. The results obtained are summarised in Tables 5 and 6. 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 train set, the obtained policy performs, on average 3.4% better than the *rolling intrinsic* policy. For the top 50% of the training days the profitability ratio is higher than 4.1% and in some cases it even exceeds 10%. Overall, the total

Table 5 Descriptive statistics of the returns obtained on the days of the train set for all considered policies

	π^{FQ}		$\pi^{APEXDQN}$		π^{HE}		π^{LA_1}		π^{LA_2}		π^{RI_g}		$\pi^{RI_{5min}}$		π^{RI}
	V (€)	r (%)	V (€)	r (%)	V (€)	r (%)	V (€)	r (%)	V (€)	r (%)	V (€)	r (%)	V (€)	r (%)	V (€)
Mean	752.2	3.4	726.6	1.4	724.6	− 2.6	832.0	12.7	839.2	14.0	727.6	− 0.1	736.3	− 0.5	746.2
Min	269.2	− 38	284.5	− 52.3	303.2	− 16	299.5	− 4.0	309.5	− 5.7	178.7	− 39.4	297.3	− 23	282.1
25%	527.0	− 2.5	515.4	− 2.6	488.8	− 7.6	569.7	4.4	570.9	5.3	511.6	− 7.8	519.2	− 5.4	507.0
50%	703.4	4.1	667.7	0.6	652.4	− 3.9	732.4	8.5	754.2	9.2	661.5	− 1.2	673.1	− 0.35	655.1
75%	857.7	11.1	811.1	7.2	802.5	− 0.7	901.1	17	912.9	18.9	824.8	6.9	812.6	4.2	813.6
Max	2958	54.7	2303	34.7	4853	37.7	4923	56	4941	76.8	2983	47.0	4916	25	4830
Sum	68,925	–	66,119	–	65,936	–	75,711	–	76,370	–	66,209	–	67,008	–	67,907

Table 6 Descriptive statistics of the returns obtained on the days of the test set for all considered policies

	π^{FQ}		$\pi^{APEXDQN}$		π^{HE}		π^{LA_1}		π^{LA_2}		π^{RI_g}		$\pi^{RI_{5min}}$		π^{RI}
	V (€)	r (%)	V (€)	r (%)	V (€)	r (%)	V (€)	r (%)	V (€)	r (%)	V (€)	r (%)	V (€)	r (%)	V (€)
Mean	667.9	3.8	669.1	3.9	616.9	− 4.2	712.6	10.6	724.0	12.2	682.3	5.5	623.0	− 2.3	645.2
Min	153.7	− 26.7	187.9	− 9.4	150.8	− 19.6	201.2	− 4.6	189.6	− 7.4	149.7	− 32.8	188.1	− 28	181.6
25%	490.9	− 0.7	501.0	0.4	459.0	− 8.2	508.0	4.2	523.0	4.8	499.0	− 4.1	459.6	− 7.4	476.5
50%	649.9	4.0	632.3	3.3	595.1	− 3.8	682.1	8.0	694.8	8.5	627.1	4.2	609.0	− 1.95	620.8
75%	814.1	9.9	772.0	7.1	715.2	− 0.8	850.7	15.0	862.7	18.9	810.8	11.7	734.9	2.6	753.3
Max	1661	40.9	1471.4	19.9	1357	32.9	1777	51.5	1834	55.5	2025	71.7	1367	27.8	1398
Sum	102,937	–	101,708	–	93,775	–	108,313	–	110,045	–	103,716	–	94,706	–	98,071

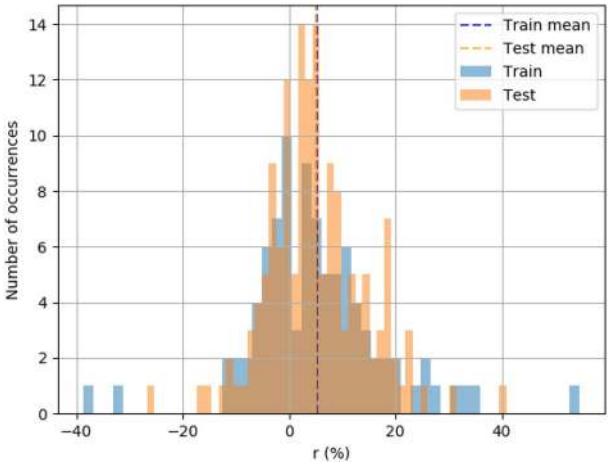

Fig. 7 Profitability ratio

profits coming from the fitted Q policy add up to €68925, yielding a difference of €1018 (1.4%) more than the profits from the *rolling intrinsic* for the set of 92 days considered.

The fitted Q policy yields on average a 3.8% 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 4% in comparison to the *rolling intrinsic*. The difference between the total profits resulting from the two policies over the set of 152 days considered amounts to €4866 (4.9%).

The distribution of training and test set samples according to the obtained profitability ratio is presented in Fig. 7. It can be observed that most samples are spread in the interval between -10% and 20% and that the distribution has a positive skew. However, as discussed earlier, the back-testing of a strategy in historical data may differ from the outcomes in real deployment for various reasons.

It can be observed that the anticipative benchmark policies π^{LA_1} and π^{LA_2} yield on average a 12% improvement with respect to the *rolling intrinsic*. This implies that the margins for performance improvements are quite tight in this particular instance that is considered. Moreover, we can see that there are cases that the look-ahead policies lead to negative profitability ratios. We remind at this point that, the *rolling intrinsic* strategy yields profits if the spread between different tradable products changes sign and if it makes sense to swap trading decisions. Instead, in the considered approach, the action to "Idle" would be desirable when there is no change in the sign between the prices of two products and the spread between these products increases in time. Furthermore, in our approach the agent would try to capture the largest spread by selecting to "Trade" at the right instant. We can observe that, given the selected trading interval of $\Delta t = 15$ min there exist several price swaps between products such that the *rolling intrinsic* is more profitable than a look-ahead policy. It is interesting to see that the *rolling intrinsic* variant $\pi^{RI_{5min}}$, where the agent selects to "Trade" at a higher frequency ($\Delta t = 5$ min), yields on average negative profitability ratios in both train (-0.5%) and test (-2.5%) sets. This indicates that $\pi^{RI_{5min}}$ is somewhat more restrictive since price swaps between products do not occur in such high frequency. The results for the variant of the *rolling intrinsic* π^{RI_g}, that selects to "Trade" only at gate closures, suggest that an agent should not start commit its unit ("Trade") since the beginning of the trading horizon, but it should wait until the first gate closure. In particular, π^{RI_g}

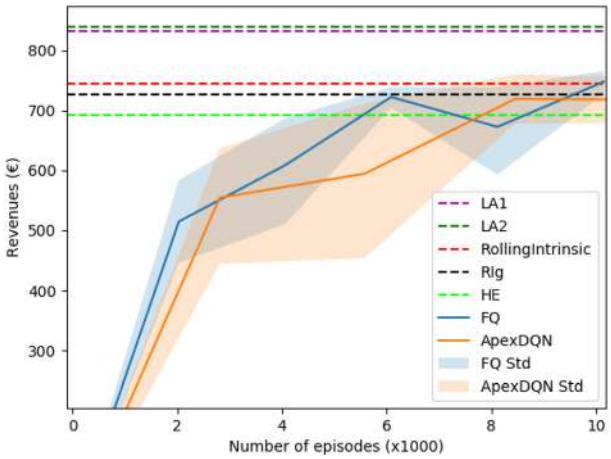

Fig. 8 Progressive evaluation in the train set

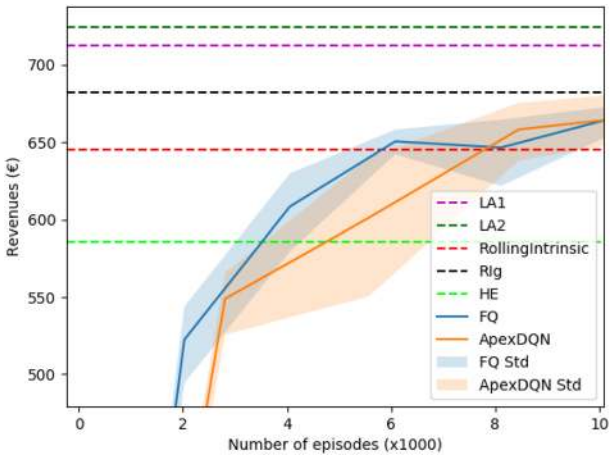

Fig. 9 Progressive evaluation in the test set

leads to similar results to π^{RI} in the train set and a 5.5% improvement in the test set, that is slightly better than the fitted Q policy π^{FQ}.

The heuristic policy π^{HE}, that solely relies on computed statistics depending on the time-step t, does not manage to outperform the *rolling intrinsic* and yields a profitability ratio of -2.6% in the train set and -4.2% in the test set. The APE-X DQN policy $\pi^{APEXDQN}$ manages to outperform the *rolling intrinsic* benchmark by 1.4% on the train set and by 4% on the test set. However, it does not reach the performance improvements achieved by the fitted Q policy.

The evolution of the expected return of the fitted Q iteration policy $V^{\pi^{FQ}}$ and the APE-X DQN policy $V^{\pi^{APEXDQN}}$ as function of the training episodes (number of trajectories collected) is presented in Figs. 8 and 9 for the train and test set respectively. We can observe that, at the early steps both learning-based methods manage to reach a performance comparable to the *rolling intrinsic*. Later in the training process, they progressively learn the right moments to idle in order to increase their returns. The shaded area in both graphs represents the variance obtained between the ten different runs.

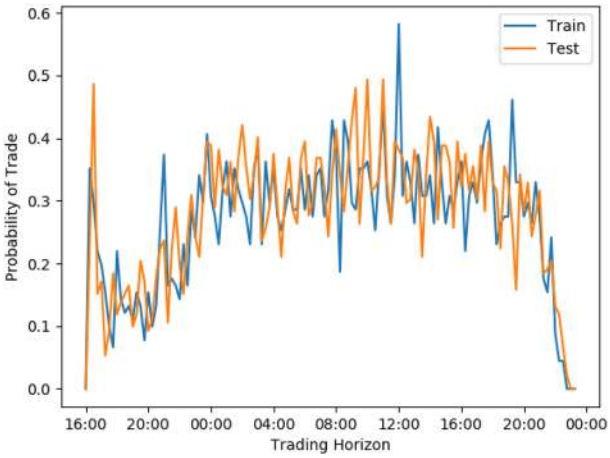

Fig. 10 Percentage of days that the "Trade" action is selected by the look-ahead policy π^{LA_1} over the trading horizon

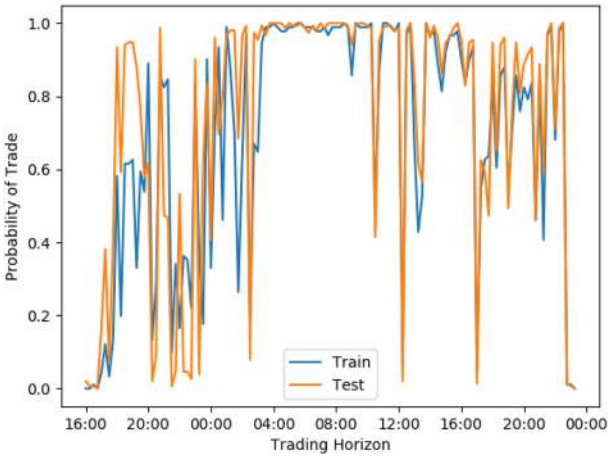

Fig. 11 Percentage of days that the "Trade" action is selected by π^{FQ} over the trading horizon

5.7 Trading policy analysis

In this section, we compare the policy obtained by the fitted Q iteration π^{FQ} with the look-ahead policy π^{LA_1}. For each time step in the trading horizon T, we compute the percentage of days for which each policy has selected the action to "Trade". The results are presented in Figs. 10 and 11 for π^{LA_1} and π^{FQ} respectively. As we can observe in Fig. 10, the look-ahead policy suggests that the agent should trade at the beginning of the trading horizon, and then wait until the gate closure of the first products. After that, the agent should mostly trade until the end of the horizon. We can see in Fig. 11 that the fitted Q agent does not trade in the beginning of the horizon, but instead starts trading approximately around 19:00 until 20:30. Then, the agent starts trading again at the first gate closure and idles only a few times until the end of the horizon. Both policies suggest that there is an upsurge of potential profits as trading approaches to the first gate closure at 23:30. This fact is supported by the increased number of orders that are on average observed at this time of the trading horizon, as shown in Fig. 12.

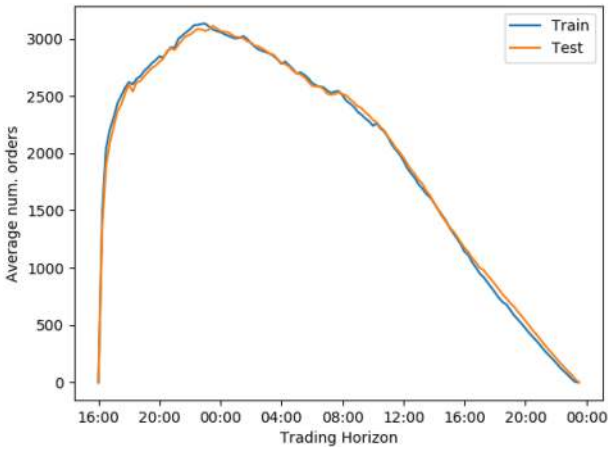

Fig. 12 Average number of orders observed by the agent over the trading horizon

5.8 Sensitivity analysis

In this section, we present a sensitivity analysis of the profit potential in the CID market with respect to the specifications of the storage unit. To this end, we apply all the previously described policies on different storage unit configurations. More specifically, we vary the charging/discharging power $\overline{G_i}/\overline{C_i}$ and the charging/discharging efficiency η. The results obtained by applying the considered policies on the test set are summarized in Table 7. It can be observed that, the look-ahead policies π^{LA_1}, π^{LA_2} achieve better performance improvements with respect to the *rolling intrinsic*, as the power capacity of the storage and the efficiency decrease. This outcome suggests that it becomes more relevant (beneficial) to idle and wait for good opportunities to arise when the storage capabilities are more restricted. Naturally, the total amount of profits collected decreases as the power capacity and efficiency decrease. The performance gain from policy π^{RI_g} remains steady at approximately 6% over all storage configurations. It is interesting to see that π^{RI_g} reaches on average half of the performance of the look-ahead policies. The reinforcement learning policies manage to outperform the *rolling intrinsic* by approximately 4% but do not succeed in reaching the π^{RI_g} performance gains. The profits of the reinforcement learning policies show a marginal increase with the decreasing power capacity and efficiency. Finally, $\pi^{RI_{5min}}$ results in a profitability ratio of approximately -2.5% for all the cases with efficiency $\eta = 0.9$. However, it yields similar performance to π^{RI} for the cases with efficiency $\eta = 0.8$.

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

Table 7 Mean values of the returns and the profitability ratios obtained on the days of the test set for all considered policies for different storage unit specifications

		π^{FQ}		$\pi^{APEXDQN}$		π^{HE}		π^{LA_1}		π^{LA_2}		π^{RI_g}		$\pi^{RI_{5min}}$		π^{RI}
$\overline{G_i}/\overline{C_i}$	η	V (€)	r (%)	V (€)	r (%)	V (€)	r (%)	V (€)	r (%)	V (€)	r (%)	V (€)	r (%)	V (€)	r (%)	V (€)
8	0.9	667.9	3.8	669.1	3.9	616.9	– 4.2	712.6	10.6	724.0	12.2	682.3	5.5	623.0	– 2.3	645.2
8	0.8	431.9	3.6	428.0	3.4	391.4	– 6.4	475.7	14.5	485.4	16.8	444.8	5.7	405.8	– 0.52	415.4
6	0.9	546.0	2.7	551.5	3.5	510.2	-4.4	594.6	11.6	604.3	13.5	567.5	6.4	513.6	– 2.5	533.0
6	0.8	353.0	0.0	361.1	4.2	317.3	– 8.1	398.1	14.8	403.2	16.2	371.6	5.6	336.2	– 0.55	346.7
4	0.9	398.7	1.5	410.4	4.7	373.7	– 4.4	440.4	12.6	447.5	14.4	417.4	6.1	379.0	– 2.5	392.6
4	0.8	263.5	1.6	268.0	4.0	238.9	– 7.6	295.9	15.4	299.2	16.7	273.9	5.2	248.5	– 0.88	256.3

i. This fact gives rise to issues related to the validity of the assumption that the process is Markovian.

We further assumed (Assumption 7) in Sect. 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 Sect. 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. In particular, the reduced pseudo-state may not 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 Action space reduction

The presented action space (High-level actions) is rather restricted in the sense that the storage unit will buy energy for a product only if it can sell it back to another product at the same instant. According to this definition of the action space, there is no risk of buying energy without using it later. However, as expected, this strategy results in reduced profits eventually. The action space reduction performed leads to a rather constrained set of admissible policies. The restrictions arise from the imposed rule that no trade of energy is allowed if it cannot be physically backed (Assumption 5). Although this assumption is made to fully comply with the German regulation, it is rather restrictive on the profits that can be achieved by the storage unit.

Alternatively, one can relax this assumption and use the reduced action space A^{red}. This would significantly increase the dimensionality of the action space and the need for exploration. Additionally, that would imply the need for a risk measure in order to quantify and control the freedom to which the resulting policy is operating.

6.4 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 Sect. 4.4 introduces further restrictions on the set of available actions. Although having a small and discrete space is convenient for the optimization 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 optimization 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 certain areas 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 obtained policy is compared against a number of benchmark strategies. The results illustrate that reinforcement learning-based policies are able to outperform on average (5%) the *rolling intrinsic* on out-of-sample data. The proposed method can serve in practice as a decision support tool to energy trading activities. Finally, the impact of the storage size on the collected revenues is evaluated. Results show that as the storage capabilities become more restricted, waiting for good opportunities to arise is more beneficial in terms of revenues for the storage operator.

The main limitations of the developed strategy originate from: (1) 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 (2) 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.

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 execution specifications that can apply to an order.
$\mathbb{B}$	Set that contains all benchmark policies.
$\hat{E}$	Number of episodes (trajectories).
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.
$\hat{N}_t(\tau)$	Set unique indices of the orders available at step t that correspond to delivery time-step τ.
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.
$\tilde{T}$	Continuous trading timeline, i.e. time interval between first possible trade and last possible trade.
T	Discretized trading timeline.
$\hat{T}(x)$	Discretised trading interval for product x.
$\check{T}$	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
$\underline{C_i}$	Minimum consumption level for the asset of agent i.
$\overline{C_i}$	Maximum 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.
$\underline{decay}$	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.
$\overline{h}$	Sequence length of past information.
$\bar{h}_{max}$	Maximum sequence length of past information.
$H_.$	Hourly products in the order book.
$HH_.$	Half-hourly products in the order book.
$I(\tau)$	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.
$p_t^{\text{“}Trade\text{"}}$	Probability to select action “Trade”.
$\underline{Q}$.	Quarter-hourly products in the order book.
$\overline{SoC_i}$	Maximum state of charge of storage device.
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.
$a'_{i,t}$	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.
$P_{i,t}^{mar}(x)$	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.
$r^{\pi^b}_{d_b}$	Profitability ratio of policy π^b at day d.
$r^{\pi^b}_{sum}$	Profitability ratio of policy π^b for the sum of returns over set.
$\hat{r}_{i,t+\psi}$	Future potential profits in ψ steps.
$s_{i,t}$	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.
$enc_h(\cdot)$	Encoder that maps from the original state space H_i to pseudo-state space Z_i.
$enc_a(\cdot)$	Encoder that maps from the high level action space A'_i to the original action space A_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.
$P_{\theta_{t,0}}(\cdot)$	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.
$V_d^{\pi^{FQ}}(\cdot)$	Return of the fitted Q policy π^{FQ} for day d.
$V_d^{\pi^{RI}}(\cdot)$	Return of the "rolling" intrinsic policy π^{RI} for day d.
V^{π^b}	Return of a benchmark policy π^b for day d.
$\mu_t(\cdot)$	Policy function at time t.
$\pi_i(\cdot)$	Policy followed by agent i.
$\pi^{LA}(\cdot)$	Look-ahead policy.
$\pi^{RI}(\cdot)$	Rolling intrinsic policy.
$\pi^{RI_g}(\cdot)$	Rolling intrinsic policy that selects to "Trade" only on gate closure.
$\pi^{RI_{5min}}(\cdot)$	Rolling intrinsic policy that selects to "Trade" every 5 min.
$\pi^{APEXDQN}(\cdot)$	Asynchronous prioritized experience replay deep Q networks (APEXDQN) policy.
$\pi^{FQ}(\cdot)$	Fitted Q policy.
$\pi^b(\cdot)$	A benchmark policy.
$\rho(\cdot)$	Trading revenue function.

Acknowledgements The authors would like to thank the insightful comments of the four anonymous reviewers that largely contributed to the final version of the manuscript.

References

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

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

Balardy, C. (2017a). 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.

Balardy, C. (2017b). 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, September 3–6, 2017*. International Association for Energy Economics.

Bertrand, G., & Papavasiliou, A. (2019). Adaptive trading in continuous intraday electricity markets for a storage unit. *IEEE Transactions on Power Systems*. https://doi.org/10.1109/TPWRS.2019.2957246.

Bertsekas, D. P. (2005). *Dynamic programming and optimal control* (Vol. 1). Belmont, MA: Athena Scientific.

Boomsma, T. K., Juul, N., & Fleten, S.-E. (2014). Bidding in sequential electricity markets: The Nordic case. *European Journal of Operational Research, 238*(3), 797–809. https://doi.org/10.1016/j.ejor.2014.04.027.

Borggrefe, F., & Neuhoff, K. (2011). Balancing and intraday market design: Options for wind integration. DIW discussion papers 1162, Berlin. http://hdl.handle.net/10419/61319.

Braun, S., & Hoffmann, R. (2016). Intraday optimization of pumped hydro power plants in the German electricity market. *Energy Procedia, 87,* 45–52. https://doi.org/10.1016/j.egypro.2015.12.356.

Busoniu, L., Babuska, R., De Schutter, B., & Ernst, D. (2017). *Reinforcement learning and dynamic programming using function approximators*. Londdon: CRC Press.

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

EPEXSPOT. EPEXSPOT Operational rules, 2019. https://www.epexspot.com/document/40170/EPEX

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

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

Garnier, E., & Madlener, R. (2015). Balancing forecast errors in continuous-trade intraday markets. *Energy Systems, 6*(3), 361–388.
Gönsch, J., & Hassler, M. (2016). Sell or store? An ADP approach to marketing renewable energy. *OR Spectrum, 38*(3), 633–660. https://doi.org/10.1007/s00291-016-0439-x.
Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press. http://www.deeplearningbook.org.
Gray, J., & Khandelwal, P. (2004). Towards a realistic gas storage model. *Commodities Now, 7*(2), 1–4.
Hagemann, S. (2015). Price determinants in the German intraday market for electricity: An empirical analysis. *Journal of Energy Markets*, *8*, 21–45.
Hassler, M. (2017). Heuristic decision rules for short-term trading of renewable energy with co-located energy storage. *Computers and Operations Research*. https://doi.org/10.1016/j.cor.2016.12.027.
Henriot, A. (2014). Market design with centralized wind power management: Handling low-predictability in intraday markets. *Energy Journal, 35*(1), 99–117. https://doi.org/10.5547/01956574.35.1.6.
Horgan, D., Quan, J., Budden, D., Barth-Maron, G., Hessel, M., Van Hasselt, H., & Silver, D. (2018). Distributed prioritized experience replay. arXiv:1803.00933.
Jiang, D. R., & Powell, W. B. (2014). Optimal hour-ahead bidding in the real-time electricity market with battery storage using approximate dynamic programming (pp. 1–28). https://doi.org/10.1287/ijoc.2015.0640, arxiv:1402.3575.
Karanfil, F., & Li, Y. (2017). The role of continuous intraday electricity markets: The integration of large-share wind power generation in Denmark. *Energy Journal, 38*(2), 107–130. https://doi.org/10.5547/01956574.38.2.fkar.
Le, H. L., Ilea, V., & Bovo, C. (2019). Integrated European intra-day electricity market: Rules, modeling and analysis. *Applied Energy, 238,* 258–273. https://doi.org/10.1016/j.apenergy.2018.12.073.
Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994* (pp. 157–163). Elsevier.
Löhndorf, N., & Wozabal, D. (2020). Gas storage valuation in incomplete markets. *European Journal of Operational Research, 288,* 318–330.
Meeus, B. L., & Schittekatte, T. (2017). The EU electricity network codes: Course text for the Florence School of Regulation online course.
Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., & Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *International conference on machine learning* (pp. 1928–1937).
Neuhoff, K., Ritter, N., Salah-Abou-El-Enien, A., & Vassilopoulos, P. (2016). Intraday markets for power: discretizing the continuous trading?.
PandŽić, H., Morales, J. M., Conejo, A. J., & Kuzle, I. (2013). Offering model for a virtual power plant based on stochastic programming. *Applied Energy, 105,* 282–292. https://doi.org/10.1016/j.apenergy.2012.12.077.
Pérez Arriaga, I., & Knittel, C., et al. (2016) *Utility of the future. An MIT energy initiative response*. ISBN 9780692808245. https://energy.mit.edu/uof.
Plazas, M. A., Conejo, A. J., & Prieto, F. J. (2005). Multimarket optimal bidding for a power producer. *IEEE Transactions on Power Systems, 20*(4), 2041–2050. https://doi.org/10.1109/TPWRS.2005.856987.
Scharff, R., & Amelin, M. (2016). Trading behaviour on the continuous intraday market Elbas. *Energy Policy, 88,* 544–557. https://doi.org/10.1016/j.enpol.2015.10.045.
Spot, N. (2018). Xbid cross-border intra day market project. https://www.nordpoolspot.com/globalassets/download-center/xbid/xbid-qa_final.pdf.
The European Commission. 2030 energy strategy, 2017. https://ec.europa.eu/energy/en/topics/energy-strategy-and-energy-union/2030-energy-strategy.
Watkins, C. J., & Dayan, P. (1992). Q-learning. *Machine learning, 8*(3–4), 279–292.