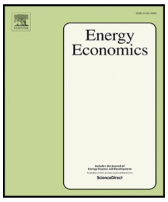
Energy Economics 149 (2025) 108673

Contents lists available at ScienceDirect

Energy Economics

journal homepage: www.elsevier.com/locate/eneeco



Reinforcement learning for bidding strategy optimization in day-ahead energy market

Luca Di Persio[a], Matteo Garbelli[a,*], Luca Maria Giordano[b]

[a] University of Verona, Department of Computer Science, Strada le Grazie 15, Verona, 37134, Italy
[b] University of Milano, Department of Mathematics, Via Cesare Saldini 50, Milano, 20133, Italy

ARTICLE INFO

JEL classification:
Q41
Q47
C57
D44
C73
N74

MSC:
91B26
37N40

Keywords:
Bidding strategy
Electricity auction
Euphemia
Day ahead energy market
Reinforcement learning

ABSTRACT

In day-ahead markets, participants submit bids specifying the amounts of energy they wish to buy or sell and the price they are prepared to pay or receive. However, the dynamic for forming the Market Clearing Price (MCP) dictated by the bidding mechanism is frequently overlooked in the literature on energy market modeling. Forecasting models usually focus on predicting the MCP rather than trying to build the optimal supply and demand curves for a given price scenario. This article develops a data-driven approach for generating optimal offering curves using Deep Deterministic Policy Gradient (DDPG), a reinforcement learning algorithm capable of handling continuous action spaces. Our model processes historical Italian electricity price data to generate stepwise offering curves that maximize profit over time. Numerical experiments demonstrate the effectiveness of our approach, with the agent achieving up to 85% of the normalized reward, i.e. the ratio between actual profit and the maximum possible revenue obtainable if all production capacity were sold at the highest feasible price. These results demonstrate that reinforcement learning can effectively capture complex temporal patterns in electricity price data without requiring explicit forecast models, providing market participants with adaptive bidding strategies that improve profit margins while accounting for production constraints.

1. Introduction

Electricity markets across Europe face unprecedented volatility due to the increasing integration of renewable energy sources, each with unique constraints related to weather dependency, production variability, and limited storage capabilities. This volatility creates both challenges and opportunities for market participants, making sophisticated bidding strategies increasingly valuable. From financial, economic, and ecological perspectives, effectively modeling market dynamics has become crucial not only for maximizing returns but also for facilitating the ongoing energy transition. Day-ahead electricity markets represent a critical component of the European power trading system. In these markets, participants submit bids specifying both quantity (q) and price per unit (p) for delivery in specific hours of the following day. The market clearing occurs at the intersection of aggregated supply and demand curves, establishing the Market Clearing Price (MCP) while balancing consumption and production. This process is managed through the EUPHEMIA algorithm (Pan-European Hybrid Electricity Market Integration Algorithm) (EUPHEMIA Public Description Single Price Coupling Algorithm, 2020), which facilitates integrated price formation across European power exchanges.

Despite extensive research on electricity price forecasting, there remains a significant gap in the literature regarding the development of bidding strategy and how energy prices influence optimal strategy. The existing approaches typically focus primarily on predicting the MCP (Nima et al., 2010; Gao et al., 2000; Bunn Derek, 2000; Li et al., 2007) rather than directly optimizing bidding curves. This research gap is particularly problematic for market participants who must make daily bidding decisions in an environment characterized by uncertainty and incomplete information where the economic impact of suboptimal bidding strategies can be substantial.

Our work addresses these limitations through several key contributions:

1. We formulate the bidding strategy problem as a stochastic optimal control problem and propose a Reinforcement Learning (RL) solution specifically tailored to electricity markets. As noted in Nima et al. (2010), the price of electricity is the most important

[*] Corresponding author.
E-mail address: matteo.garbelli@univr.it (M. Garbelli).

https://doi.org/10.1016/j.eneco.2025.108673
Received 11 July 2024; Received in revised form 6 May 2025; Accepted 8 June 2025
Available online 7 July 2025
0140-9883/© 2025 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

signal to all market participants. We use this signal to drive our RL algorithm.

2. Unlike previous approaches using discrete action spaces, see e.g. Xiong et al. (2002), we employ Deep Deterministic Policy Gradient (DDPG) to handle the continuous, high-dimensional action space of offering curves.
3. We directly incorporate historical prices, using real-world data from the Italian electricity market, into the state representation, allowing the agent to learn complex temporal patterns without requiring explicit price forecast models.

By using a model-free RL approach, our method enables market participants to optimize bidding strategies without explicit knowledge of the market clearing mechanism or competitors' behavior. Instead, the agent (energy operator) interacts with a stochastic environment, selecting offering curves based on historical price patterns and receiving rewards based on the resulting profit. Our objective is to compute a deterministic, offline optimal policy that maximizes cumulative discounted rewards given historical price states.

A key motivation for this single-agent framework lies in its scalability and adaptability to more complex, interactive settings. While this work focuses on an individual agent operating in a partially observable environment, the proposed algorithm serves as a critical building block for future extensions to Markov games, where multiple strategic sellers and buyers dynamically interact in competitive or collaborative bidding scenarios. By first isolating and resolving the challenges of learning robust policies under price uncertainty and partial market information in the single-agent case, we establish a foundation for analyzing equilibrium behaviors and decentralized learning dynamics in multi-agent systems. We refer to Gronauer and Diepold (2022), Lee et al. (2020), Tampuu et al. (2017) for comprehensive surveys of multi-agent reinforcement learning techniques that extend single-agent frameworks to handle strategic interactions among multiple participants in competitive markets.

The paper is organized as follows: we conclude Section 1 by a review of RL methodology and its application to energy market problems; in Section 2 we introduce the theoretical setting for stating the electricity auction problem as an optimal control one; in Section 3 we present the Deep Deterministic Policy Gradient (DDPG) method and its adaptation to the electricity auction framework; in Section 4, we present details of the implementation scheme, e.g. the description of the data and the choice of the hyperparameters; in Section 5 we present the results of the numerical simulations and report some considerations and limitations of the algorithm. We conclude the article with Section 6 sketching future directions that may employ this algorithm as a reference starting point.

Literature review. RL (Sutton and Barto, 2018; Jaimungal, 2022) is a learning paradigm that maps situations to actions to maximize a numerical reward signal through repeated experience gained by interacting with the environment. The agent aims to develop a strategy that maximizes the expected cumulative reward over time by learning a policy that maps states to actions. Some of the most common algorithms for RL rely on learning optimal action-value functions by computing the corresponding Q-value, i.e. the quality, the optimal expected future value of the selected action given a particular space. Reference works for Q-learning are contained in Watkins and Dayan (1992). For its extension, the Deep Q-Networks (DQN), we refer to Mnih et al. (2015). A further step is introduced by Actor-criticism methods, which combine the advantages of policy gradient methods and value function approximation to improve the learning process. The actor is responsible for generating actions based on the current policy, while the critic learns to evaluate the policy by estimating the value function. We refer to Konda and Tsitsiklis (2000) for a complete discussion.

DDPG is an off-policy algorithm that extends the idea of the actor-critic method to continuous action spaces (Lillicrap et al., 2015). DDPG uses a deep neural network to approximate the policy and another deep Neural Network (NN) to approximate the value function. Throughout our research, we use the RL model developed in Lillicrap et al. (2015) as the reference algorithm for our setting by developing its adaptation for the setting considered. Following this track, DDPG has been applied to learn optimal bidding strategies for generators and energy storage systems in day-ahead markets and real-time markets (Ye et al., 2020). Focusing on other projects that applied RL algorithms to the energy field, we start by citing (Xiong et al., 2002): the authors model the electricity auction market using a Q-learning algorithm considering each supplier bidding strategy as a Markov Decision Problem where the agents learn from experience an optimal bidding strategy to maximize its payoff. Although there are certain limits in terms of application — in the case studies considered in Xiong et al. (2002) such as the use of simple synthetic datasets as well as discrete Q-tables for the pairings of actions-state, this work serves as a landmark for the research developed in this article. The main differences with the problem studied in Xiong et al. (2002) rely on the source of data since we choose to employ historical times series rather than using synthetic data such as in Xiong et al. (2002) as well as the development of a more complex RL model. Q-learning has always been used in electricity auctions to learn bidding strategies for market participants, such as generators and retailers (Nicolaisen et al., 2001). However, the discrete nature of Q-learning can limit its applicability to auctions with large or continuous state and action spaces. In contrast, policy gradient methods can handle continuous state and action spaces, making them suitable for electricity auctions with complex market dynamics. One limitation of policy gradient methods is that they may require a large batch of samples for stable learning.

Recent developments have demonstrated the effectiveness of reinforcement learning in optimizing bidding strategies for day-ahead electricity markets. For instance, graph-enhanced deep RL methods have been shown to improve bid coordination in multi-agent settings (Weng et al., 2025). Other studies have explored the integration of storage with renewables using advanced actor-critic algorithms such as TD3 (Cardo-Miota et al., 2025), or ensured operational safety constraints via safe RL policies (Rokhforoz et al., 2023). The strategic behavior of price-making participants like Virtual Power Plants has also been tackled using multi-agent RL approaches such as MATD3 (Jiang et al., 2024).

Outside the applications to the day-ahead market, we can find, e.g. Gajjar et al. (2003), where actor-critic methods are applied for learning bidding strategies and demand response for regulating the power-exchange bidding mechanism in deregulated power system management, offering a balance between exploration and exploitation. In Bâra et al. (2024), an AI-powered approach is developed to forecast electricity prices for driving the optimization of trading strategies for electricity market players across both day-ahead and balancing markets. The authors propose a novel two-step forecasting method: first predicting the imbalance sign (deficit/surplus) using classification algorithms, then incorporating this prediction as a feature to forecast actual electricity prices. A similar approach for the optimization of trading strategies is studied in Bâra and Oprea (2025) where an Energy Assistant is developed to help prosumers (consumers who also produce energy) in Local Electricity Markets. Several integrated algorithms are considered, from predictive analysis to the optimization of bidding prices and quantities via RL. More generally, we refer to the survey (Di Persio et al., 2024) for a complete overview and best practice of ML use-cases for electricity markets.

2. Problem statement and model formulation

Modern energy networks rely heavily on day-ahead electricity markets, which effectively offer a framework to purchase electricity for next-day delivery. Throughout the paper, we focus our attention on the Italian electricity market. The unitary MCP of energy, also known as PUN (*Prezzo Unitario Nazionale*) in the Italian market, is established by

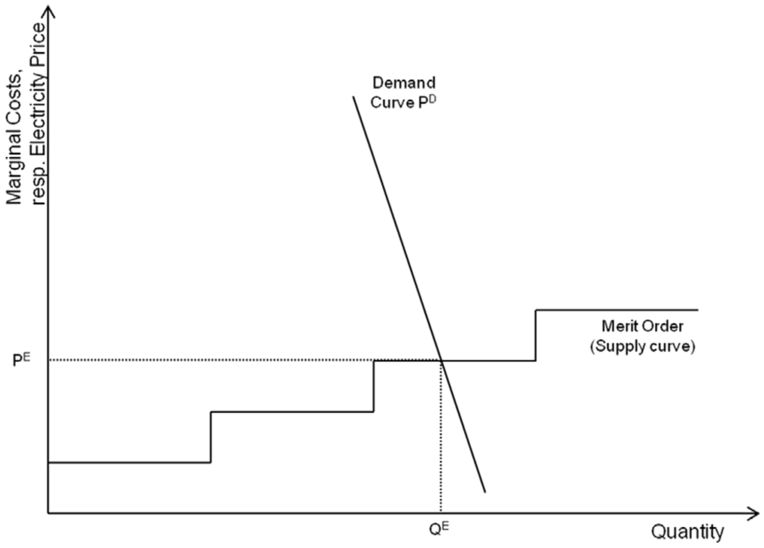


Fig. 1. Offering Curve: the intersection with the demand (thus the reward) is computed with the next-day price of electricity.

a bidding procedure involving several market participants, including producers, consumers, and traders. PUN represents the cost of producing the last unit of electricity needed to meet demand. The Euphemia algorithm sets it (EUPHEMIA Public Description Single Price Coupling Algorithm, 2020) as the intersection of the supply and demand curves.

A critical issue for electricity suppliers is how to optimally bid on the auction market to maximize their profit. We model the optimal control problem into a single-agent RL setting, solved via the DDPG algorithm, introduced in Lillicrap et al. (2015), that uses deep NNs to approximate policy and value functions in a high-dimensional, continuous action space.

2.1. Stochastic optimal control for electricity auction problem

The producer operates in a stochastic environment for selecting the best bidding strategy, striving to maximize their profit over the long term while meeting the needs of the available resources. At each stage t, given a state $s_t \in S$, the seller selects and executes an $a_t \in A$ that, in our setting, corresponds to a stepwise energy/price function, depending on the learned policy $\pi : S \to P(A)$. The agent's goal is to take actions that will maximize its expected long-term performance with an unknown transition function P. To achieve this, the agent learns a behavior policy $\pi : S \to P(A)$ that optimizes its expected performance in the long run. The system progresses from state s_t under joint action $a_t \in A$, based on the transition probability function P, to the next state s_{t+1}, providing updated information on the aggregated load, corresponding unit loads and the new unit price.

In Fig. 1, we report an example of an offering curve that we obtain corresponding to an action that the agent can take at time t.

The reward function $r_t : S \times A \times S \to \mathbb{R}$ corresponds to the received feedback signal transitioning from (s_t, a_t) to s_{t+1}. Since the immediate reward is insufficient for providing insights into the long-term profit, it is crucial to introduce the return value R_t, defined over a finite time horizon T. The following expression gives the return value R_t:

$$
\begin{aligned}
R_t &= r_{t+1}(s_t, a_t) + \gamma^{t+1} r_{t+2}(s_{t+1}, a_{t+1}) + \dots + \gamma^{T-1} r_T(s_{T-1}, a_{T-1}) = \\
&= r_{t+1}(s_t, a_t) + \sum_{i=t+1}^{T-1} \gamma^i r_{i+1}(s_i, a_i)\,. \qquad (1)
\end{aligned}
$$

In Eq. (1), $\gamma^i \in [0,1]$ corresponds to the discount factors that determine the importance of future rewards compared to the immediate ones, with lower values focusing on short-term rewards. Hence, the return value R_t corresponds to the discounted sum of future rewards, allowing agents to optimize their actions for long-term profit. Each agent receives R_{t+1} as immediate feedback for the state transition. Hence, the agent aims to optimize an objective corresponding to the return value R_t

$$
J = \mathbb{E}_{r_t, s_t, a_t \sim \pi}[R_t]\,, \qquad (2)
$$

That corresponds to learning a policy that maximizes the cumulative future payoff to be received starting from any given time t until the terminal time T.

The agent's value function associated with such a control problem reads.

$$
V(s_t) = \max_{a_t \in A} \mathbb{E}\left[R(s_t, a_t, s_{t+1})\right]\,. \qquad (3)
$$

The dynamic programming principle implies that V satisfies the Bellman equation.

$$
V(s_t) = \max_{a_t \in A} \mathbb{E}\left[r(s_t, a_t, s_{t+1}) + \gamma V(s_{t+1})\right]\,. \qquad (4)
$$

In RL, it is useful to define an action-value function to measure the 'quality' of taking a specific action a, and hence is called the Q function $Q : S \times A \to \mathbb{R}$ defined as

$$
Q(s_t, a_t) = R(s_t, a_t) + \gamma \max_{a \in A} Q(s_{t+1}, a_t)\,. \qquad (5)
$$

The Q-function solves the Bellman equation by recursively updating the value of taking a specific action a_t in a given state $_t$

$$
Q(s_t, a_t) = R(s_t, a_t) + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1})\,. \qquad (6)
$$

By iteratively applying this update, the Q-function converges to the optimal action-value function, effectively solving the Bellman equation. This process allows the algorithm to learn the optimal bidding strategy by estimating the quality of different offering curves based on historical price data. We build the critic network to approximate Eq. (6) in Section 3. .

2.2. Construction of the model

Following the paradigm of a repeated day-ahead electricity auction market, the agent will attempt to maximize its profit in the long run in a recursive way.

At each time step t, the agent receives an observation consisting in of 24-hour PUN array prices PUN_t of the last d days, see Fig. 2. We assume a fully observable environment where the state of the environment is represented by the market electricity prices expressed in €/MWh.

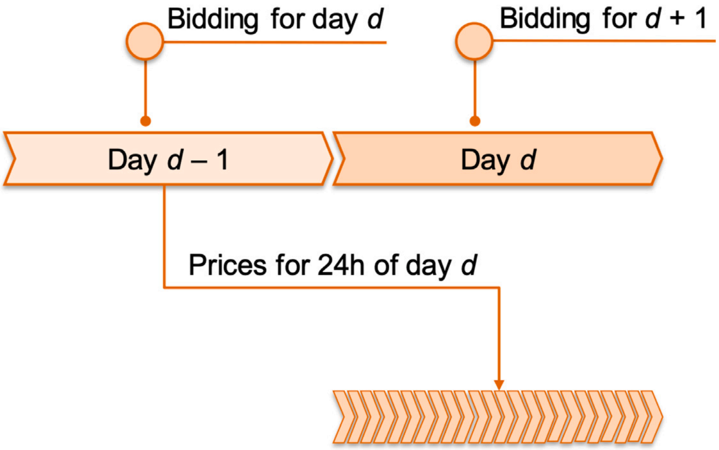


Fig. 2. Bidding settlement in day-ahead auctions (from Maciejowska et al. (2022))

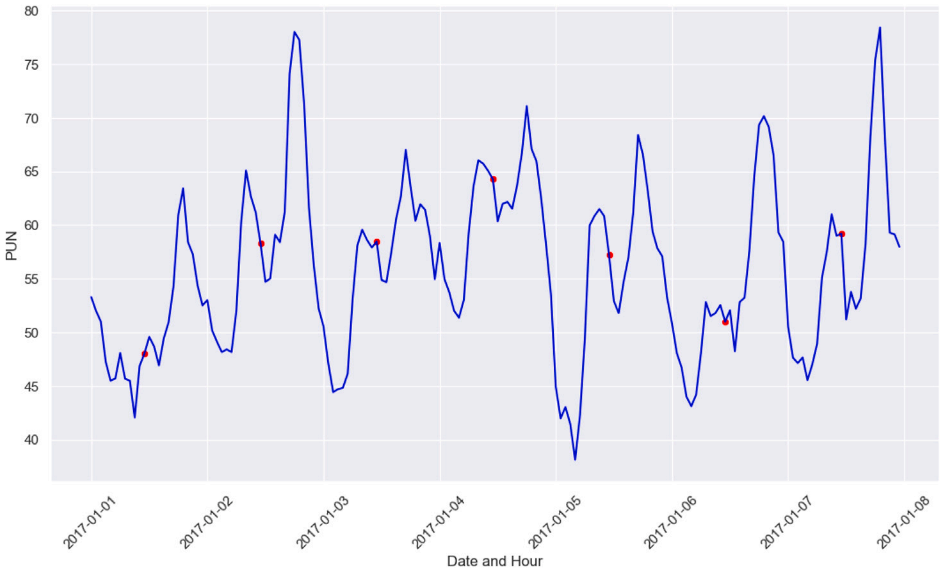


Fig. 3. A batch of 7 days electricity prices describes the state of the RL algorithm. The red dots correspond to the specific hour (12 AM) we want to build the optimal curves.

We briefly describe how state, actions, and reward are defined for this problem.

State. The current state s_t of the system is given by:

- *Electricity prices*. An extracted 7-day batch of historical electricity prices is denoted as PUN_t, represented by a matrix of 168 values, 24 hours times 7 days, shown in Fig. 3.
  For each mode of production k corresponding to K different sources, we have:
- *Unitary Production Costs*: The production cost C_t^k for each of the K production modes at time t.
- *Maximum Dispatched Volumes* D_t^k: The maximum volume that can be produced for each of the K production modes at time t.

Action. The actions correspond to offering curves like the one depicted in Fig. 1. The steps are described by a couple of quantities/prices (V_i, P_i) the supplier intended to purchase. The number of steps I the curves are constructed is a hyper-parameter of the model. Each *step* i characterized by: - *Volumes* V_i: The volumes offered for sale. - *Prices* P_i: The prices at which these volumes are offered. In Fig. 4, we anticipate the space offering curve (we set $I = 3$) obtained as outputs of the Actor-Network (see Fig. 3).

The curve of the Actor NN is obtained as the offering that minimizes the reward (in terms of obtained profit) given the batch of prices for the last 7 days.

Reward. The reward is computed by considering the volumes for which the offered price P_i is below the market clearing price PUN_{t+1}. The revenue from the accepted offers $(P_i \cdot V_i)$ is calculated, and then the production cost $(C^i \cdot V_i)$ is subtracted, resulting in the final reward.

Considering K modes of production and I bidding, i.e. steps for the offering curve, at each hour t, the reward r_t is computed as:

$$r_t(PUN_{t+1}, P_i, V_i, C_k, D_k) = \sum_{k=1}^{K} \sum_{i=1}^{I} \left(P_i \cdot V_i - C_k \cdot V_i\right) \mathbb{1}_{\{P_i \leq PUN_{t+1}\}} \qquad (7)$$

Where:

- PUN_{t+1} is the market clearing price for the same hour of the next day.
- P_i is the price offered at bidding i.
- V_i is the volume offered at price P_i.
- C_k is the unitary production cost for mode k.
- D_k is the maximum producible volume for production mode k.
- $\mathbb{1}_{\{P_i \leq PUN_{t+1}\}}$ is the indicator function that equals 1 if the offered price P_i is less than or equal to PUN_{t+1}, and 0 otherwise.

The presence of the indicator function ensures that the reward is computed by considering the revenue from the accepted offers (where

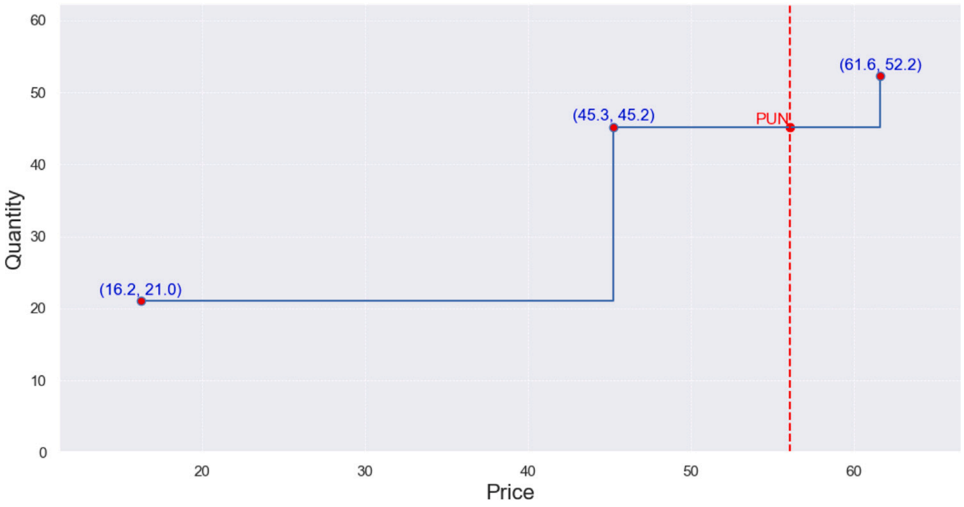


Fig. 4. Offering curve corresponding to an output of the Actor NN. The reward is computed with Eq. (7) using only the accepted offers, i.e. the ones with an offered price less than the registered pun (the red vertical line in the image).

$P_i \leq PUN_{t+1}$). According to Eq. (7), the reward corresponds to the realized profit, computed by subtracting to the daily gain the production cost C_k for the sold volumes V_i.

We refer to the next section for a detailed study of how these quantities are merged into the DDPG algorithm, as well as other numerical simplifications introduced in the model.

3. The DDPG algorithm for electricity auction

DDPG is a model-free actor-critic algorithm based on the deterministic policy gradient with continuous action spaces. We provide a complete graphical summary of the developed algorithm in Fig. 5.

3.1. The DDPG algorithm

In general terms, the DDPG algorithm consists of the following steps:

1. Initialize the actor network with weights θ_μ and the critic network with weights θ_Q.
2. Initialize the target networks for the actor and critic with the same weights: $\theta'_\mu = \theta_\mu$ and $\theta'_Q = \theta_Q$.
3. Sample a minibatch of transitions (s_t, a_t, r_t, s_{t+1}) from the replay buffer.
4. Update the critic network by minimizing the loss:
$$L = \frac{1}{N}\sum_t (y_t - Q(s_t, a_t|\theta_Q))^2 \tag{8}$$
where $y_t = r_t + \gamma Q(s_{t+1}, \mu(s_{t+1}|\theta'_\mu)|\theta'_Q)$ is the target Q-value, and μ is the deterministic policy from the actor network. The critic loss (8) corresponds to the Mean Squared Error (MSE) between the predicted Q-values and the actual rewards plus the discounted Q-value of the next state (i.e., the Bellman equation (6)). This loss guides the critic network to approximate the true Q-values (5) as accurately as possible.
5. Update the actor-network using the sampled policy gradient introduced in Eq. (13);
6. Update the target networks using the soft update rule:
$$\theta'_\mu \leftarrow \tau\theta_\mu + (1-\tau)\theta'_\mu, \quad \theta'_Q \leftarrow \tau\theta_Q + (1-\tau)\theta'_Q, \tag{9}$$
where $\tau \ll 1$ is a small constant that controls the update rate.

An exploration strategy based on noise processes, e.g., the Ornstein–Uhlenbeck one, adds temporally correlated noise to the actions. Accordingly, the noise process X_t is defined by the following SDE:

$$dX_t = -\theta(X_t - \mu)dt + \sigma dW_t, \tag{10}$$

that we discretize by the following Euler–Maruyama method:

$$X_{t+\Delta t} = X_t - \theta(X_t - \mu)\Delta t + \sigma\sqrt{\Delta t}\xi_t \tag{11}$$

Where ξ_t represents a random sample from a standard normal distribution.

Finally, the generated noise is added to the actions produced by the Actor-network:

$$a_t = \mu(s_t|\theta_\mu) + X_t \tag{12}$$

where a_t is the action taken at time t, $\mu(s_t|\theta_\mu)$ is the action produced by the actor-network for the state s_t, and X_t is the noise generated by the Ornstein–Uhlenbeck process.

3.2. Adaptation of the DDPG algorithm for electricity auction

We consider a single agent setting of an energy operator that interacts with the market (environment) in discrete time steps.

In summary, at each time step t, the agent:

1. receives an observation x_t consisting in of 24-hour PUN array prices P_t of the last d days.
2. generates an action a_t, i.e. a stepwise curve modeling the offering curve corresponding to the output of the actor-network;
3. observes a feedback scalar reward r_t corresponding to the agent's payoff.

Both actor and critic are approximated using deep Feed Forward Neural Networks (FFNN) with a second set of target FFNNs. In Fig. 6, we sketch the Feed Forward NN we use to model the Actor Network.

The actor is updated by applying the chain rule using the sampled policy gradient.

$$\nabla_{\theta_\mu} J \sim \frac{1}{N}\sum_t \nabla_a Q(s, a|\theta_Q)\Big|_{\{s=s_t, a=\mu(s_t)\}} \nabla_{\theta_\mu}\mu(s \mid \theta_\mu)\Big|_{\{s=s_t\}} \tag{13}$$

that minimizes the distance between the current policy's actions and actions that maximize expected rewards. This technique estimates the gradient of the expected cumulative reward concerning the policy parameters using samples collected during interactions with the environment.

The Critic Network approximates the Q function for a given state–action pair. On the other hand, the critic function $Q(s, a)$ approximates the Q-value value function given a pair (price-offering curve) that estimates the expected return by approximating the Bellman equation (6), as illustrated in Fig. 7.

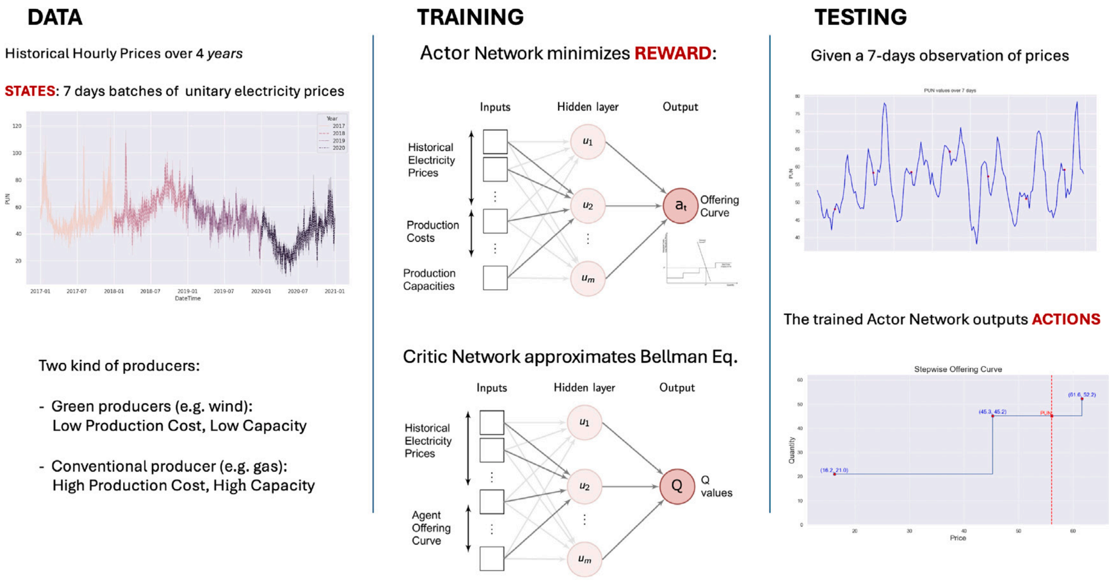

DATA
Historical Hourly Prices over 4 years
STATES: 7 days batches of unitary electricity prices
Year
2017
2018
2019
2020
PUN
DateTime
Two kind of producers:
- Green producers (e.g. wind): Low Production Cost, Low Capacity
- Conventional producer (e.g. gas): High Production Cost, High Capacity
TRAINING
Actor Network minimizes REWARD:
Inputs
Hidden layer
Output
Historical Electricity Prices
Production Costs
Production Capacities
u_1
u_2
u_m
a_t
Offering Curve
Critic Network approximates Bellman Eq.
Inputs
Hidden layer
Output
Historical Electricity Prices
Agent Offering Curve
u_1
u_2
u_m
Q
Q values
TESTING
Given a 7-days observation of prices
PUN values over 7 days
PUN
The trained Actor Network outputs ACTIONS
Stepwise Offering Curve
(16.2, 21.0)
(45.3, 45.2)
(51.6, 52.2)
PUN
Quantity
Price

Fig. 5. Graphical summary of the RL data-driven model.

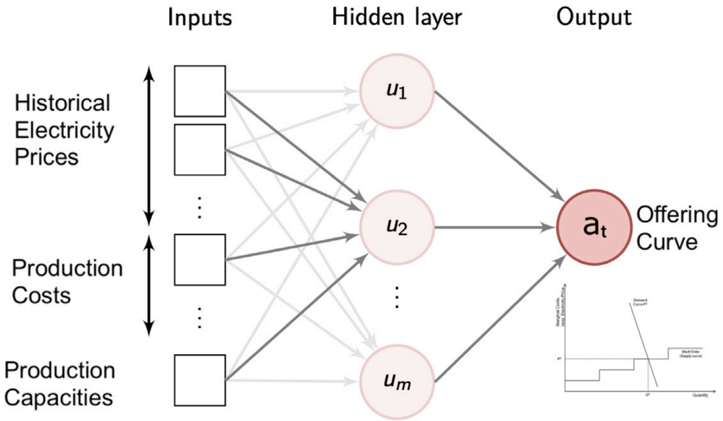


Fig. 6. The Feed Forward Actor Network produces a vectorial output representing the Offering Curve.

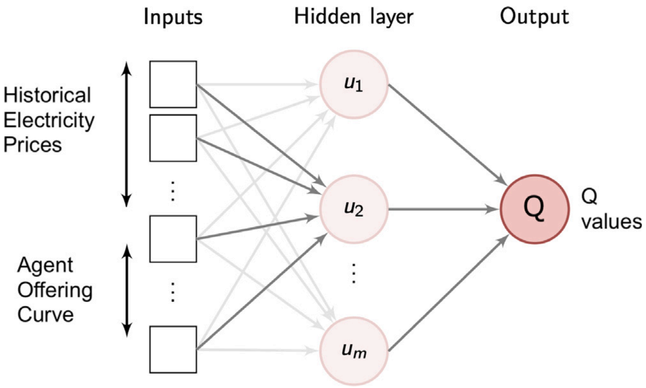


Fig. 7. The Feed Forward Critic Network approximates the Bellman Equation given (s_t, a_t).

Table 1
DDPG hyperparameters.

Episodes	1000, 1500
Length of an Episode [Days]	5, 6, 10
Batch Size	64
Hidden Size	64
Actor Learning Rate	0.0001
Critic Learning Rate	0.00001
Discount Factor	0.99
Tau	0.01
Max Memory Size	50,000

4. Implementation of the algorithm

4.1. Description of the dataset

The dataset consists of Italian electricity prices over 4 years. Precisely, we use hourly data for PUN from 01/2017 to 12/2020, plotted in Fig. 8.

We use stratified sampling to ensure that training and testing sets have a representative mix of data throughout the timeframe.

Concerning the quantities introduced in Section 2.2, we assume curves composed of $I = 3$ bidding couples (p, q). We set $K = 3$ modes of production and $I = 3$ fixed generation capacities and different production costs corresponding to different energy sources. We consider a constant array of production costs and available power to calibrate our result to calculate a value for the reward function defined in Eq. (7). In particular, we set the number of sources of production K to 3 and set the cost $C^i = [10, 30, 60]$ and the production capacity $D^i = [30, 200, 800]$. This choice, from a modeling point of view, the source with a low marginal cost and low capacity, i.e. $(10, 30)$, represents a renewable source of production, the one with a high marginal cost and capacity a conventional one (e.g. gas) plus an intermediate one. Without loss of generality, one could also consider stochastic quantities, with the additional effort of storing the ad-hoc Stochastic Differential Equation (SDE) simulation while adding an extra source of randomness to the NN input. We decide to leave this additional feature for future work.

Another hypothesis we introduce is that the agent only knows about its expenses, available resources, historical electricity prices and nothing about its competitors. As a result, its bidding strategy can be represented as a stochastic process that adheres to a decision-making framework.

Concerning the development algorithm, we report some tools we used in the DDPG method to improve the learning process's stability:

- a replay buffer is employed to store past experiences and sample mini-batches of transitions (the so-called experiences arrays) for training;
- using different learning rates for actor and critic networks. The target networks are updated slowly, using a soft update rule with a small mixing factor, which helps to stabilize learning;
- Adding regularization techniques, such as L2 regularization, to the loss functions for the actor and critic networks helps prevent overfitting;
- carefully tuning the hyperparameters, such as the learning rates, discount factor, and soft update rate, significantly impacts the performance of DDPG. We conduct a systematic search or optimization techniques to find the best hyperparameters reported in Table 1.

Besides working in mini-batches, we add common noise to randomize the actions. We assume noise as a discretized Ornstein–Uhlenbeck process as defined in Eq. (11). By an empirical calibration, we set the value of the rate of mean reversion θ to 0.15, the mean μ to 1 with diffusion σ taking values in the interval $[1, 10]$ guaranteeing an efficient impact for the update of the action (12).

Consequently, we derive the following scheme:

Algorithm 1 DDPG for electricity auctions

```
1: Initialize the Actor network $\mu(s_t|\theta_\mu)$ and the Critic network $Q(s_t, a_t|\theta_Q)$ with random weights $\theta_\mu$ and $\theta_Q$.
2: Initialize the target networks $\mu'(s_t|\theta'_\mu)$ and $Q'(s_t, a_t|\theta'_Q)$ with weights $\theta'_\mu \leftarrow \theta_\mu$ and $\theta'_Q \leftarrow \theta_Q$.
3: Initialize the Ornstein–Uhlenbeck noise process $X_t$.
4: For each episode:
       Initialize the environment and obtain the initial state $s_0$.
       For each time step $t$:
           1: Select the action $a_t = \mu(s_t|\theta_\mu) + X_t$, where $X_t$ is the noise generated by the Ornstein–Uhlenbeck process.
           2: Execute the action $a_t$ in the environment and observe the reward $r_t$ and the next state $s_{t+1}$.
           3: Store the transition $(s_t, a_t, r_t, s_{t+1})$ in the replay buffer.
           4: Update the noise process $X_{t+\Delta t} = X_t - \theta(X_t - \mu)\Delta t + \sigma\sqrt{\Delta t}\xi_t$, where $\xi_t$ is a random sample from a standard normal distribution.
           5: If the replay buffer contains enough samples, sample a mini-batch of transitions $(s_j, a_j, r_j, s_{j+1})$ from the replay buffer.
           6: Update the Critic network by minimizing the loss:
              $L(\theta_Q) = \frac{1}{m}\sum_{j=1}^{m}\left(Q(s_j, a_j|\theta_Q) - (r_j + \gamma Q'(s_{j+1}, \mu'(s_{j+1}|\theta'_\mu)|\theta'_Q))\right)^2$   (14)
           7: Update the Actor Network using the sampled policy gradient:
              $\nabla_{\theta_\mu} J(\theta_\mu) \approx \frac{1}{m}\sum_{j=1}^{m} \nabla_a Q(s_j, a|\theta_Q)\Big|_{a=\mu(s_j|\theta_\mu)} \nabla_{\theta_\mu}\mu(s_j|\theta_\mu)$   (15)
           8: Update the target networks using soft updates:
              $\theta'_\mu \leftarrow \tau\theta_\mu + (1-\tau)\theta'_\mu, \quad \theta'_Q \leftarrow \tau\theta_Q + (1-\tau)\theta'_Q,$   (16)
9: Repeat until the desired level of performance is achieved or a maximum number of episodes is reached.
```

4.2. Selection of the DDPG hyperparameters

We briefly describe the hyperparameters we use for the DDPG algorithm while referring to Table 1 for the reference values we set for the implementation.

An episode means dealing with a complete sequence of interactions between the agent and the environment. The *Episodes* corresponds to the times the agent will engage with the environment to learn and improve its policy. The *Length of an Episode* refers to the number of time steps or interactions the agent experiences within a single episode. It defines how long the agent operates in the environment before the episode concludes. The *Batch Size* refers to the number of experiences (state action-reward-next state tuples) sampled from the replay buffer at each iteration of the training process. The *Hidden Size* refers to the number of neurons or units in the hidden layers of the neural networks used in the DDPG algorithm. The *Actor Learning Rate* and the *Critic Learning Rate* control step size of the gradient update at which the actor network's weights θ_μ and the critic network's weights θ_Q are updated during training. The *Discount Factor* parameter γ introduced in Eq. (1) represents the relative importance of future rewards compared to immediate rewards. A higher gamma value places more importance on long-term rewards, potentially encouraging the agent to consider the future consequences of its actions. The *Target Update Factor* τ is a relaxation factor that defines how often the parameters are copied from the original networks to the target for the network parameters

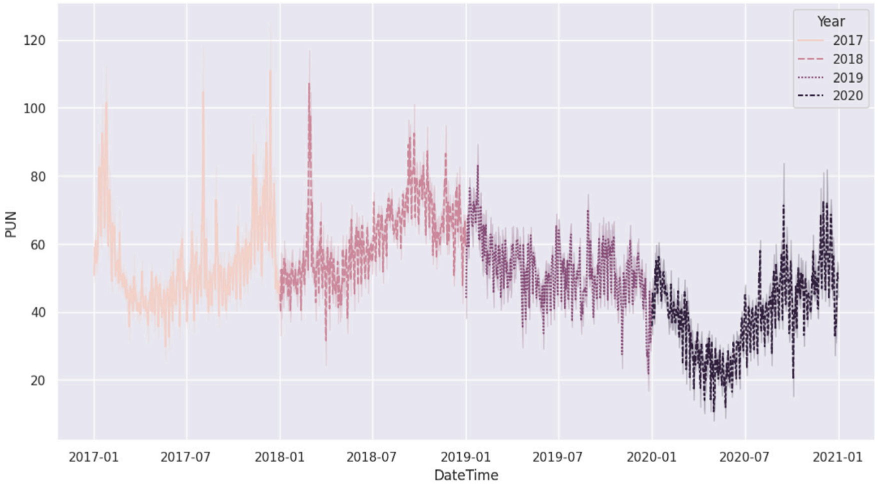


Fig. 8. Italian hourly PUN from 01-Jan-2017 to 31-Dec-2020 for a total of 35040 data points downloaded from https://www.mercatoelettrico.org/it/Download/DatiStorici.aspx.

copied. The *Max Memory Size* determines the capacity of the replay buffer, which is a crucial component in DDPG. The replay buffer stores past experiences (state, action, reward, next state) that the agent uses to learn from.

Since DDPG hyperparameters are interconnected, finding the correct tuning significantly impacts the algorithm's performance and stability. We calibrate the values of Table 1 by an experimental procedure by adjusting them to optimize the global reward. We refer to the following Subsection for a complete overview of the tuning procedure and some technical considerations.

5. Numerical results

Rewards obtained from the environment might vary significantly since they depend on the price time series that is highly not stationary, as we can see from Fig. 8. To address this issue, we include a normalization factor into the reward to consider the potential *normalized reward* that can be obtained each time t.

The normalized reward, denoted as $\mathcal{R}_{norm}$, is calculated by dividing the actual reward r introduced in Eq. (7) for a given time step by the maximum possible value of the reward r_{max} that can be achieved under the particular time step conditions. When production costs, capacity, and the matching pun for a specific time step are considered, the maximum reward reflects the most profit that may be realized. For more steady and practical learning, this normalization scales the reward values to a consistent range between 0 and 1: when the agent receives the maximum reward, the normalized reward will be 1. The normalized reward ranges from 0 to 1 if the agent performs below the maximum. This option enables the agent to focus on tailoring its strategy in response to the relative performance improvement, which improves the consistency of comparisons between various learning contexts.

Besides the normalized reward, we plot two interesting metrics: policy loss and critical loss. The agent learns to improve its policy by adjusting the weights of the actor-network to maximize the expected cumulative reward J, defined in the equation. (2). The policy loss measures the discrepancy between the actions chosen by the current policy and those that would lead to higher expected rewards. It is approximated by the sampled policy gradient introduced in (13). The Critic Loss introduced in Eq. (8) is associated with training the critic network, measures the precision of the predictions of the Q-value of the critic and guides the critic network to approximate the Q-values that satisfy the Bellman equation (6).

5.1. Learning performance

We observe that the algorithm is sensitive to the initialization of parameters such as number of episodes, episode length (in days), and production cost values, which must be carefully chosen to achieve optimal performance.

For this reason, we perform several simulations for different values of *Episodes* and *Length of an Episode*, reporting the obtained results for 3 different configurations with episode lengths of 5, 7 and 10 days respectively, all with a total of 1000 episodes.

In all plots, we can see how the policy loss consistently decreases over time; hence, the actor moves towards a better policy. In contrast, the critic loss shows an initial spike before gradually decreasing. Thus, the Q-value estimate improves only after approximately 300 *Episodes.* This behavior is a typical pattern in the training dynamics of RL algorithms that can be attributed to the interaction with a stochastic environment. Specifically, in the early stages of training, the agent's policy might be far from optimal, leading to higher Q-values and a more considerable critic loss. As the agent explores the environment and gathers more experience, it gradually refines its policy, causing the loss of the critic to decrease. Moreover, this delayed learning is also linked to the target networks, which are updated slowly, leading to a more significant initial critical loss. The critic loss decreases as training progresses and the target networks catch up.

However, the gradual decrease in policy and critic losses as training metrics and the increase of the normalized reward indicate the agent's convergence towards more optimal policies.

5.2. Considerations and challenges

As previously encountered in other settings such as (Rashedi et al., 2016), the market rewards may be highly stochastic in the single-agent framework. They may be unable to converge to a fixed point, leading to oscillatory behaviors of market producers. This behavior is expected because generators do not account for the strategies of their competitors, which may evolve over time. In contrast, the gradual decline in policy loss and the varying patterns in critic loss are signs of the DDPG algorithm's learning process. They demonstrate the agent's evolution from initial exploration and poorly predicted Q-values to more refined value estimations and a more developed strategy. These patterns indicate that the algorithm is learning from its training data and adapting to its environment, which are generally expected behaviors in reinforcement learning.

A notable distinction between our configurations lies in the behavior of the critic loss. In the 5-day episode configuration (Fig. 9), we observe

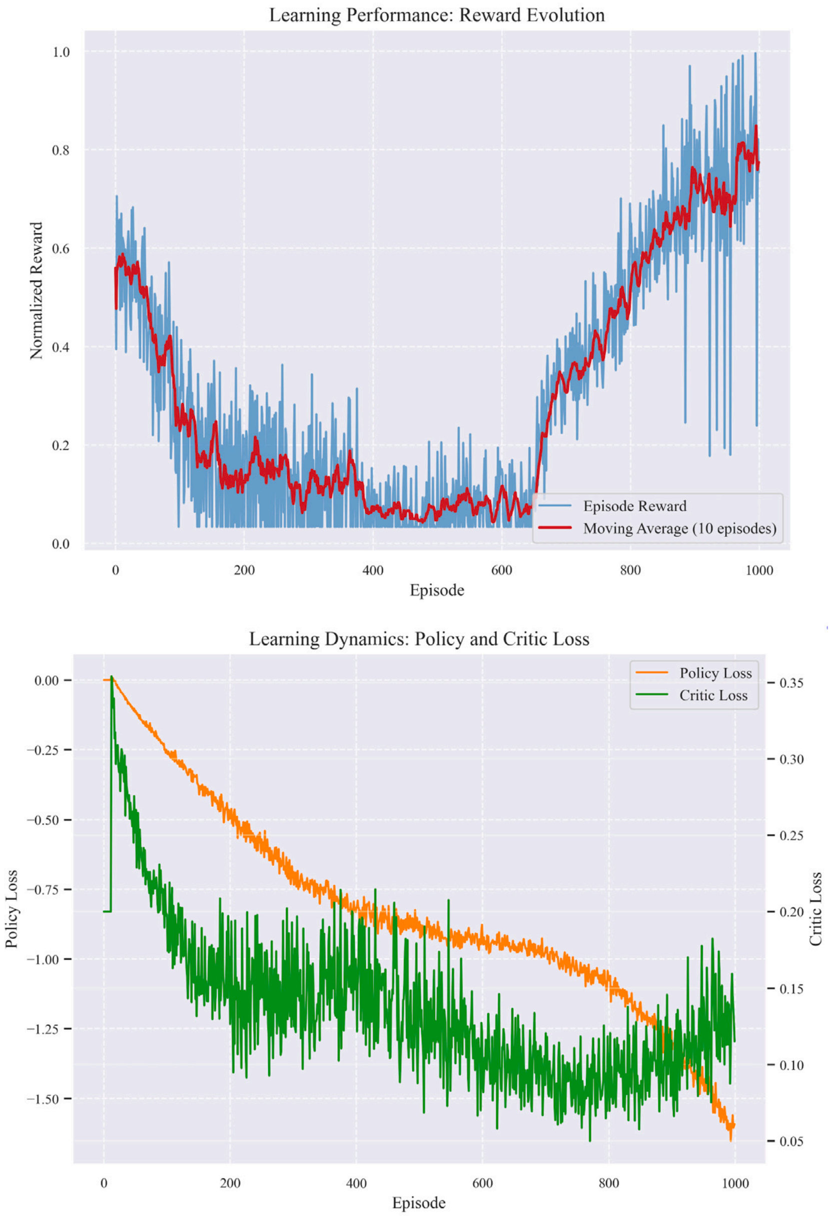


Fig. 9. **Simulation 1**: Episodes = 1000, length of episode = 5 days.

that the critic loss initially spikes, then consistently decreases as training progresses. This pattern suggests that the critic network gradually improves its Q-value predictions, resulting in more accurate value estimations over time. However, in the 7-day and 10-day configurations (Figs. 10 and 11), we observe a different pattern where the critic loss initially decreases but then begins to increase or shows higher volatility in later episodes.

This varying critic loss phenomenon across different episode lengths is not necessarily indicative of poor performance. Increasing critic loss can occur even as the overall performance improves, which is a known phenomenon in deep reinforcement learning. Several explanations for this behavior include:

1. Non-stationarity of the target: As the policy improves, the distribution of states encountered by the agent changes, making the critic's job increasingly difficult as it must adapt to new state distributions.

2. Exploration–exploitation dynamics: Longer episodes may lead to more diverse state explorations, challenging the critic to make accurate predictions across a wider range of states.

3. Moving target problem: In actor-critic methods, both networks are trained simultaneously, creating a continuous adjustment process where the critic attempts to evaluate an evolving policy.

4. Inherent volatility in market price data: The high variance and complex patterns in electricity prices may create situations where precise Q-value estimation becomes increasingly challenging as the agent encounters new price scenarios.

Despite the varying critic loss patterns across configurations, we observe that the normalized reward continues to improve, and the policy loss consistently decreases across all setups. This suggests that the agent's overall performance is enhancing even when the critic's predictions become more challenging to calibrate.

The difficulties we encounter while developing the algorithm can be divided into two categories: those caused by the structure of electrical auction markets and those specifically related to the RL algorithm. Market-related challenges include partial observability of the action space (the agent can learn about competitors' actions only through historical price time series), non-stationarity of the data, and high problem dimensionality. RL-specific challenges arise from the stability

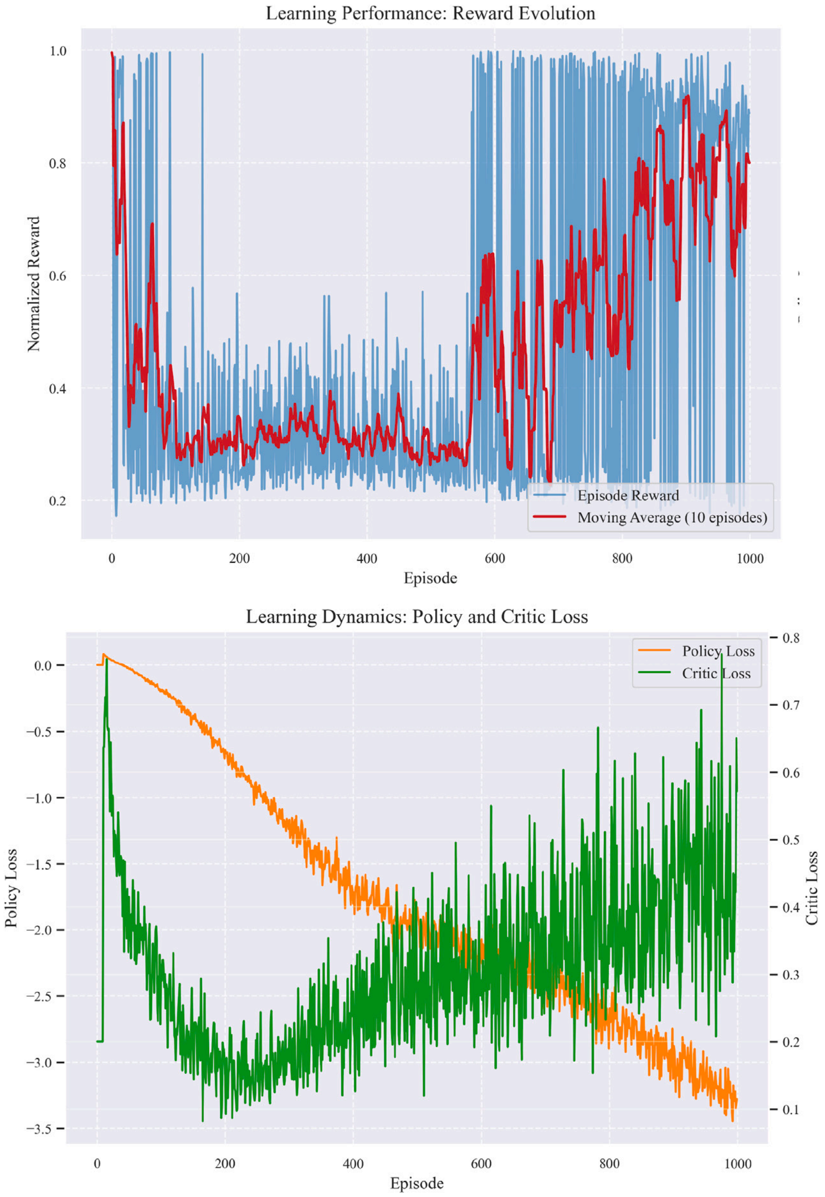


Fig. 10. **Simulation 2**: Episodes = 1000, length of episode = 7 days.

requirements of DDPG, including function approximation errors, non-stationary targets, and the complex interplay between actor and critic networks.

Additionally, when we monitor the algorithm's process, we notice several key observations related to the stability of the DDPG technique and the importance of hyperparameter selection:

- A higher number of episodes allows the agent to explore the environment more extensively, potentially leading to better policy convergence. Episodes provide opportunities for the agent to encounter a diverse range of states and price profiles, which is crucial for learning a robust policy that performs well across different scenarios. Conversely, the learning process often exhibits diminishing returns with increasing episodes. Initially, the agent might rapidly learn and improve its policy, but over time, the rate of improvement might slow down as it explores less novel situations. Summarizing, the length of an episode appears to be a crucial parameter for the exploration–exploitation trade-off;
- Also *Length of an Episode* hyperparameter can influence the agent's learning dynamics and exploration strategy. A more extended episode provides the agent more time to investigate the surroundings thoroughly and make more choices, which might result in further exploration. A shorter episode, on the other hand, would encourage the agent to focus more on using what it already knows, thereby limiting exploration. We identify a crucial value of 15 days for *Length of an Episode* below which we do not have convergence;
- By experimentation, we set the learning rate of the critic lower than the one of the actor-network. Updates to obtain more stable training by preventing one network from significantly outpacing the other;
- A larger batch size can lead to more efficient learning as it allows the agent to learn from more experiences in parallel. However, we note that employing a huge batch size may increase the amount of noise and volatility in the learning process, resulting in less consistent training. As each update is based on a smaller subset of the overall experiences, we selected a batch size that is relatively modest to encourage the agent to explore more varied encounters.

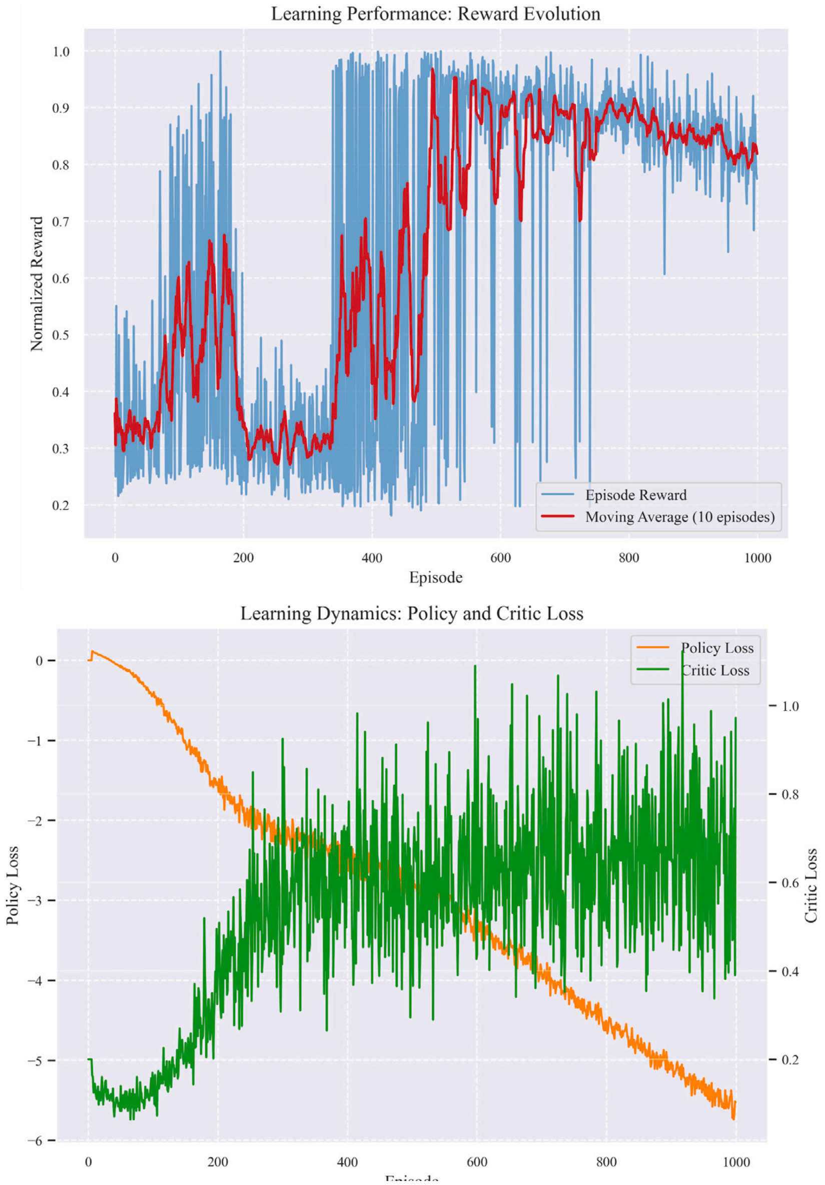


Fig. 11. **Simulation 3**: Episodes = 1000, length of episode = 10 days.

5.3. Limitations of the model

Throughout this work, we have made several assumptions and simplifications to make the complex electricity market bidding problem tractable using reinforcement learning. Here, we provide a systematic recap of these assumptions and discuss their implications as limitations of our approach.

Model. Our most significant simplification is the adoption of a single-agent framework, where the agent (electricity supplier) operates in isolation without explicitly modeling the behavior of competitors. The agent can only infer competitor actions indirectly through historical price patterns. This approach ignores the game-theoretic nature of electricity markets, where multiple suppliers strategically interact and adapt to each other's bidding strategies. In real markets, price formation emerges from these complex interactions, which our model does not capture directly.

Production capacity and costs. We simplified the production model by assuming: three distinct production sources ($K = 3$); fixed generation capacities ($D_i = [30, 200, 800]$); constant production costs ($C_i = [10, 30, 60]$).

In reality, production capacities and costs fluctuate due to fuel price volatility, maintenance schedules, and especially for renewable sources, weather conditions. Our model does not account for these dynamic factors, which can significantly impact optimal bidding strategies.

Limited bidding structure. Our offering curves consist of only three steps ($I = 3$), representing a simplified version of real market bidding structures, which typically allow for more sophisticated curve shapes with more price-quantity pairs. This limitation reduces the flexibility of the bidding strategy and potentially constrains the agent's ability to optimize profit across different market conditions.

Reward function. Our reward function, as defined in Eq. (7), only considers the immediate profit from accepted bids without accounting for start-up and shut-down costs, minimum operating levels, transmission constraints, locational factors, etc.

Stationarity. The DDPG algorithm implicitly assumes some degree of stationarity in the environment, but electricity markets exhibit non-stationary behavior due to evolving regulations, changing market participant strategies, and shifts in supply–demand fundamentals over time. Our approach may struggle to adapt to structural changes in the market that were not represented in the training data.

Neural network architecture. We employ standard feed-forward neural networks for both actor and critic components. However, the time-series nature of electricity price data might be better captured by recurrent architectures like LSTMs or attention-based models that can identify temporal patterns more effectively.

Market impact. Our model does not consider the market impact of the agent's actions. In reality, especially for large suppliers, bidding strategies can influence market clearing prices, creating a feedback loop not captured in our framework.

6. Conclusions and possible future directions

The article's goal concerns the development of an optimizing strategy for a single-agent RL setting. We develop a DDPG algorithm to learn a deterministic policy through actor-critic architecture using a Q-value function to grade the proposed offering curve. This combination allows DDPG to effectively handle continuous action spaces, making it a robust algorithm for various applications.

We extend the result in Xiong et al. (2002) by including an ML algorithm, precisely the DDPG method, and by directly using the evolution of historical prices for decision-making tasks.

Moving from the limitations highlighted in the previous section, we conclude the article by presenting some extensions of this model that may be worth further investigating: we solely test feed-forward NNs for implementing this model. We think the algorithm's performance could be enhanced by choosing architectures that are better suited for time series, including recurrent NNs, such as Long Short Term Memory NNs; we model production cost and production capacity as deterministic constant quantities. Incorporating stochastic capacities to model random fluctuations associated with renewable energy production could provide significant benefits.

This research serves as a foundational building block for developing algorithms for multi-agent or Mean Field systems, for example, by integrating the empirical distribution of the offering curves of other operators. Thereafter, the construction of a distributed optimization system – an algorithm based on decentralized coordination, such as, e.g., local rewards or consensus schemes – is the foundation for extending a single agent into a multi-agent framework.

List of variables

We recall all the variables appearing in the manuscript:
J: objective function to be maximized
s_t: state at time t
$V(s_t)$: value function at state s_t
π: policy mapping states to actions
a_t: action at time t, representing the offering curve
P: transition probability function
r_t: reward at time t
R_t: return value at time t
γ: discount factor for future rewards
$Q(s_t, a_t)$: action-value function for state s_t and action a_t
PUN_t: electricity price (Prezzo Unitario Nazionale) at time t
C_k: unitary production cost for mode k
D_k: maximum producible volume for production mode k
V_i: volume offered at bidding step i
P_i: price offered at bidding step i
I: number of steps in the offering curve
θ_μ: actor network weights
θ_Q: critic network weights
τ: soft update parameter for target networks
X_t: Ornstein–Uhlenbeck noise process
θ: mean reversion rate of Ornstein–Uhlenbeck process
μ: mean of Ornstein–Uhlenbeck process
σ: diffusion rate of Ornstein–Uhlenbeck process
θ'_μ: target actor network weights
θ'_Q: target critic network weights
K: number of production modes
y_t: target Q-value
L: loss function for the critic network
ξ_t: random sample from standard normal distribution

List of acronyms

RL Reinforcement Learning

MCP Market Clearing Price

DDPG Deep Deterministic Policy Gradient

EUPHEMIA Pan-European Hybrid Electricity Market Integration Algorithm

DQN Deep Q-Networks

NN Neural Network

PUN Prezzo Unitario Nazionale (National Unitary Price)

MSE Mean Squared Error

FFNN Feed Forward Neural Network

CV Cleared Volume

SDE Stochastic Differential Equation

LSTM Long Short-Term Memory

ML Machine Learning

AI Artificial Intelligence

CRediT authorship contribution statement

Luca Di Persio: Writing – review & editing, Supervision, Project administration, Formal analysis. **Matteo Garbelli:** Writing – original draft, Visualization, Validation, Supervision, Software, Methodology, Investigation, Formal analysis, Conceptualization. **Luca Maria Giordano:** Writing – original draft, Validation, Software, Investigation, Formal analysis, Data curation.

References

Bâra, A., Oprea, S.-V., 2025. Energy assistants for prosumers to enable trading strategies on local electricity markets. Knowl.-Based Syst. (ISSN: 0950-7051) 309, 112927. http://dx.doi.org/10.1016/j.knosys.2024.112927.

Bâra, A., Oprea, S.-V., Ciurea, C.-E., 2024. Improving the strategies of the market players using an AI-powered price forecast for electricity market. Technol. Econ. Dev. Econ. 30 (1), 312–337. http://dx.doi.org/10.3846/tede.2023.20251.

Bunn Derek, W., 2000. Forecasting loads and prices in competitive power markets. Proc. IEEE 88 (2), 163–169.

Cardo-Miota, Javier, Beltran, Hector, Pérez, Emilio, Khadem, Shafi, Bahloul, Mohamed, 2025. Deep reinforcement learning-based strategy for maximizing returns from renewable energy and energy storage systems in multi-electricity markets. Appl. Energy (ISSN: 0306-2619) 388, 125561. http://dx.doi.org/10.1016/j.apenergy.2025.125561.

Di Persio, L., Alruqimi, M., Garbelli, M., 2024. Stochastic approaches to energy markets: From stochastic differential equations to mean field games and neural network modeling. Energies 17 (23), 6106. http://dx.doi.org/10.3390/en17236106.

EUPHEMIA Public Description Single Price Coupling Algorithm, 2020. Available online from: https://www.nordpoolgroup.com/globalassets/download-center/single-day-ahead-coupling/euphemia-public-description.pdf.

Gajjar, G.R., Khaparde, S.A., Nagaraju, P., Soman, S.A., 2003. Application of an actor-critic learning algorithm for optimal bidding problem of a Genco. IEEE Trans. Power Syst. 18 (1), 11–18. http://dx.doi.org/10.1109/TPWRS.2002.807041.

Gao, Feng, Guan, Xiaohong, Cao, Xi-Ren, Papalexopoulos, A., 2000. Forecasting power market clearing price and quantity using a neural network method. In: 2000 Power Engineering Society Summer Meeting (Cat. No. 00CH37134). Seattle, WA, USA, pp. 2183–2188. http://dx.doi.org/10.1109/PESS.2000.866984.

Gronauer, S., Diepold, K., 2022. Multiagent deep reinforcement learning: a survey. Artif. Intell. Rev. 55, 895–943. http://dx.doi.org/10.1007/s10462-021-09996-w.

Jaimungal, S., 2022. Reinforcement learning and stochastic optimization. Finance Stoch. 26, 103–129. http://dx.doi.org/10.1007/s00780-021-00467-2.

Jiang, Yuzheng, Dong, Jun, Huang, Hexiang, 2024. Optimal bidding strategy for the price-maker virtual power plant in the day-ahead market based on multi-agent twin delayed deep deterministic policy gradient algorithm. Energy (ISSN: 0360-5442) 306, 132388.

Konda, V.R., Tsitsiklis, J.N., 2000. Actor-critic algorithms. In: Advances in Neural Information Processing Systems. pp. 1008–1014.

Lee, D., He, N., Kamalaruban, P., Cevher, V., 2020. Optimization for reinforcement learning: From a single agent to cooperative agents. IEEE Signal Process. Mag. 37 (3), 123–135. http://dx.doi.org/10.1109/MSP.2020.2976000.

Li, G., Liu, C.-C., Mattson, C., Lawarree, J., 2007. Day-ahead electricity price forecasting in a grid environment. IEEE Trans. Power Syst. 22 (1), 266–274. http://dx.doi.org/10.1109/TPWRS.2006.887893.

Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., 2015. Continuous control with deep reinforcement learning. Comput. Sci. 8 (6).

Maciejowska, K., Uniejewski, B., Weron, R., 2022. Forecasting Electricity Prices. Oxford research encyclopedia of economics and finance.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Hassabis, D., 2015. Human-level control through deep reinforcement learning. Nature 518 (7540), 529–533.

Nicolaisen, T., Petrov, V., Tesfatsion, L., 2001. Market power and efficiency in a computational electricity market with discriminatory double-auction pricing. IEEE Trans. Evol. Comput. 5 (5), 504–523.

Nima, Amjady, Daraeepour, A., Keynia, F., 2010. Day-ahead electricity price forecasting by modified relief algorithm and hybrid neural network. IET Gener. Transm. Distrib. 4 (3), 432–444.

Rashedi, N., Tajeddini, M.A., Kebriaei, H., 2016. Markov game approach for multiagent competitive bidding strategies in the electricity market. IET Gener. Transm. Distrib. 10, 3756–3763. http://dx.doi.org/10.1049/iet-gtd.2016.0075.

Rokhforoz, Pegah, Montazeri, Mina, Fink, Olga, 2023. Multi-agent reinforcement learning with graph convolutional neural networks for optimal bidding strategies of generation units in electricity markets. Expert Syst. Appl. (ISSN: 0957-4174) 225, 120010. http://dx.doi.org/10.1016/j.eswa.2023.120010.

Sutton, R.S., Barto, A.G., 2018. Reinforcement Learning: An Introduction;. MIT Press, Cambridge, MA, USA.

Tampuu, A., Matiisen, T., Kodelja, D., Kuzovkin, I., Korjus, K., Aru, J., et al., 2017. Multiagent cooperation and competition with deep reinforcement learning. PLoS ONE 12 (4), e0172395. http://dx.doi.org/10.1371/journal.pone.0172395.

Watkins, C.J., Dayan, P., 1992. Q-learning. Mach. Learn. 8 (3–4), 279–292.

Weng, Haoen, Hu, Yongli, Liang, Min, Xi, Jiayang, Yin, Baocai, 2025. Optimizing bidding strategy in electricity market based on graph convolutional neural network and deep reinforcement learning. Appl. Energy (ISSN: 0306-2619) 380, 124978, http://dx.doi.org/10.1016/j.apenergy.2024.124978, http://dx.doi.org/10.1016/j.energy.2024.132388.

Xiong, G., Hashiyama, S., Okuma, T., 2002. An electricity supplier bidding strategy through Q-learning. IEEE Power Eng. Soc. Summer Meet. 3, 1516–1521. http://dx.doi.org/10.1109/PESS.2002.1043645.

Ye, Y., Qiu, D., Sun, M., Papadaskalopoulos, D., Strbac, G., 2020. Deep reinforcement learning for strategic bidding in electricity markets. IEEE Trans. Smart Grid 11 (2), 1343–1355. http://dx.doi.org/10.1109/TSG.2019.2936142.