RESEARCH

Finding Individual Strategies for Storage Units in Electricity Market Models using Deep Reinforcement Learning

Nick Harder[1*], Anke Weidlich[1] and Philipp Staudt[2]

*Correspondence: nick.harder@inatech.uni-freiburg.de
[1]Institute for Sustainable Systems Engineering, University of Freiburg, Freiburg im Breisgau, Germany
Full list of author information is available at the end of the article

Abstract

Modeling energy storage units realistically is challenging as their decision-making is not governed by a marginal cost pricing strategy but relies on expected electricity prices. Existing electricity market models often use centralized rule-based bidding or global optimization approaches, which may not accurately capture the competitive behavior of market participants. To address this issue, we present a novel method using multi-agent deep reinforcement learning to model individual strategies in electricity market models. We demonstrate the practical applicability of our approach using a detailed model of the German wholesale electricity market with a complete fleet of pumped hydro energy storage units represented as learning agents. We compare the results to widely used modeling approaches and demonstrate that the proposed method performs well and can accurately represent the competitive behavior of market participants. To understand the benefits of using reinforcement learning, we analyze overall profits, aggregated dispatch, and individual behavior of energy storage units. The proposed method can improve the accuracy and realism of electricity market modeling and help policymakers make informed decisions for future market designs and policies.

Keywords: agent-based modeling; electricity markets; energy storage; multi-agent reinforcement learning; reinforcement learning

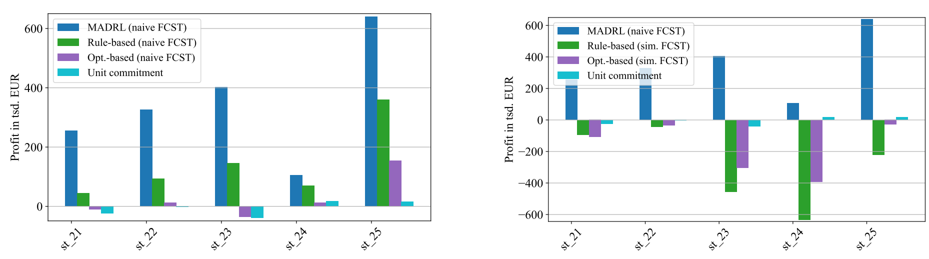
Supplementary Material

Algorithm 1 MATD3 algorithm for bidding of N energy storage units in an electricity market

Initialize critic $\theta_{1,2}$ and actor ϕ for each learning agent
Initialize target critic $\theta'_{1,2}$ and actor ϕ' for each learning agent
for episode $\in [1, M]$ **do**
  Initialize market simulation
  for t $\in [1, T]$ **do**
    Create a global observation o_t^{global}
    for i $\in [1, N]$ **do**
      Derive individual observation $o_{t,i}$ using the global and local observations
      Perform an action $a_{t,i} = \pi_{\phi_i}(o_{t,i}) + \epsilon$
      Derive the bid $B_{t,i} = (P_{t,i}, ep_{t,i})$ using the action $a_{t,i}$
    end for
    Perform the market clearing
    for i $\in [1, N]$ **do**
      Get accepted capacity $P_{t,i}^{\text{conf.}}$ and market clearing price M_t
      Calculate individual reward $R_{t,i}$
      Store transition $(s_{t,i}, a_{t,i}, r_{t,i}, s_{t+1,i})$ into the replay buffer
    end for
    if remainder $\frac{t}{\text{train.freq.}}$ is 0 **then**
      for step $\in [1, \text{grad.steps}]$ **do**
        for i $\in [1, N]$ **do**
          Sample a minibatch of $k \in B$ samples from the replay buffer
          Calculate target actions: $\tilde{\mathcal{A}}_{k,i} = \pi_{\phi'}(o'_{k,i}) + \epsilon$
          Calculate the target value: $y_{k,i} = r_{k,i} + \gamma \min_{j=1,2} Q_{\theta'_{i,j}}(\mathcal{O}'_{k,i}, \tilde{\mathcal{A}}'_{k,i})$
          Update the critics using gradient ascent: $L(\theta_{i,j}) = \frac{1}{B}\sum_{k=1}^{B}[y_{k,i} - Q_{\theta_{j,i}}(\mathcal{O}_{k,i}, \mathcal{A}_{k,i})]^2$
          if remainder $\frac{\text{step}}{\text{delay}}$ is 0 **then**
            Update the actor: $\nabla_\phi J(\phi_i) = \frac{1}{B}\sum_{k=1}^{B} \nabla_\phi \pi_{\phi_i}(o_{k,i}) \nabla_a Q_{\theta_{i,1}} \cdot \left(\mathcal{O}_{k,i}, \mathcal{A}_{k,i}^{\mathcal{A}\setminus[i]}, \pi_{\phi_i}(o_{k,i})\right)$
            Perform soft-updates:
            $\theta'_{i,j} = (1-\tau)\theta'_{i,j} + \tau\theta_{i,j}$ for $j = 1, 2$
            $\phi'_i = (1-\tau)\phi'_i + \tau\phi_i$
          end if
        end for
      end for
    end if
  end for
end for

Table 1: Actor and critic NN parameters and learning hyper-parameters.

Parameter	**Value**	**Parameter**	**Value**
Critic NN architecture	MLP, (x_input, 512, 256, 128, 1)	Training frequency	1000
Actor NN architecture	MLP, (x_input, 256, 128, 1)	Gradient steps	1000
Critic activations	(ReLU, ReLU, ReLU, Linear)	Buffer size	$5 \cdot 10^5$
Actor activations	(RELU, RELU, Tanh)	Policy delay, d	2
Observation size	128	Soft-update, τ	0.05
Action size	2	Target noise σ	0.2
Optimizer, learning rate	Adam, 10^{-4}	Target noise clip, c	0.5
Batch size, B	256	Learning starts	5
Reward discount, γ	0.999	Action noise	Gaussian



(a) using naive forecast (b) using simulated forecast

Figure 1: Total profits in Case 2 using MADRL, rule-based and optimization-based bidding strategies, and unit commitment model for units 21-25.