Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 52-4 (2019) 488–493

Building day-ahead bidding functions for seasonal storage systems: A reinforcement learning approach

Jesus Lago [*,**] Ecem Sogancioglu [***] Gowri Suryanarayana [**]
Fjo De Ridder [****] Bart De Schutter [*]

[*] Delft Center for Systems and Control, Delft University of Technology, Delft, The Netherlands (e-mail: j.lagogarcia@tudelft.nl)
[**] Algorithms, Modeling, and Optimization, VITO-Energyville, Genk, Belgium
[***] Diagnostic Image Analysis Group, Radboud University Medical Center, Nijmegen, The Netherlands
[****] Kenniscentrum Energie, Thomas More, Geel, Belgium

Abstract: Due to the increasing integration of renewable sources in the electrical grid, electricity generation is expected to become more uncertain. In this context, seasonal thermal energy storage systems (STESSs) are key to shift the delivery of renewable energy sources and tackle their uncertainty problems. In this paper, we propose an optimal controller for STESSs that, using reinforcement learning, builds bidding functions for the day-ahead market. In detail, considering that there is an uncertain energy demand that the STESS has to satisfy, the controller buys energy in the day-ahead market so that the uncertain demand is satisfied while the profits are maximized. Since prices are low during periods of large renewable energy generation (and vice versa), maximizing the profit of a STESS indirectly shifts the delivery of renewable energy to periods of high energy demand while reducing their uncertainty problems. To evaluate the proposed algorithm, we consider a real STESS providing different yearly-demand levels; then, we compare the performance of the controller to the theoretical upper bound, i.e. the optimal cost of buying energy given perfect knowledge of the demand and prices. The results indicate that the proposed controller performs reasonably well: despite the large uncertainty in prices and demand, the proposed controller obtains 70%-50% of the maximum gains given by the theoretical bound.

Keywords: Seasonal Storage, Bidding Functions, Reinforcement Learning, Energy Storage

1. INTRODUCTION

In recent years, as more renewable energy sources have been integrated into the electrical grid, storing energy has become a key aspect. Particularly, as renewable generation is uncertain and weather dependent, larger integration leads to more frequent and severe imbalances between production and consumption. As a result, as these imbalance become more frequent and severe, the management of the grid becomes more complex (Lago et al. (2018)). One of the solutions to keep the grid stable and to mitigate the negative effects of these imbalances is to use *seasonal thermal energy storage systems (STESSs)* (Xu et al. (2014)) to shift the delivery of renewable energy: by storing renewable energy during periods of positive imbalances, i.e. when generation is larger than consumption, and selling it during periods of negative imbalances, STESSs can reduce these problems. In addition, as positive imbalances usually lead to low market prices (and vice versa) (Lago et al. (2019)), shifting the energy delivery to reduce imbalances is aligned with the economic objective of STESSs, i.e. by reducing imbalances the profit of the system is also increased.

Based on these premises, it becomes clear that, to maximize the benefits of STESSs, a control algorithm that optimizes their profits is highly desirable. In particular, the control algorithm that buys electricity for the STESS needs to quantify price and demand uncertainty and then use this information to build bidding functions for the electricity market that ensure that, while the energy demand is satisfied, the cost of buying energy is minimized.

1.1 *Building bidding functions in electricity markets*

Constructing bidding functions for the electricity market is a topic that has been widely researched. While traditionally the focus has been towards the case of generator companies (see Gong et al. (2011) for a review), building bidding functions for retailers has also been studied. In particular, in the context of retailers, Fleten and Pettersen (2005) proposed one of the earliest methods: using a stochastic linear optimization problem, a method to build piecewise linear bidding functions for the day-ahead market was proposed and tested in the Norwegian market. Similarly, Carrion et al. (2007) proposed a method for building piecewise constant bidding functions for the Spanish futures market by solving a mixed-integer linear stochastic optimization problem. In a different study, Herranz et al. (2012) proposed a non-linear and non-differentiable optimization problem that constructed bidding functions for several Spanish spot markets.

While these methods work well for traditional retailers, it can be argued that there is a key difference between a traditional retailer and a STESS that makes the existing methods not suitable for the latter case. In detail, a traditional retailer buys electricity on an hourly basis to provide the requested demand (Fleten and Pettersen (2005); Herranz et al. (2012)). The main tool to maximize its profit is to hedge in different electricity markets and to incentivize consumers to provide flexible demand. By contrast, because of the seasonality and long-term aspects of STESSs, these can spend days or weeks without buying any energy and then buy 10-100 times their average demand within a few hours or days. As a result, the time scales that a traditional retailer and a STESS have to deal with are very different. As an example, if we consider the day-ahead market, a traditional retailer would likely consider the prices during the next day; however, a STESS will need to forecast the prices not only for the next day but also for the next months.

1.2 Motivation and contributions

Due to this difference in the time scales, the methods proposed in the literature are not suitable for STESS:

- The number of variables that a STESS needs to consider is much larger than that of a retailer. As the proposed methods in the literature are based on stochastic optimization, they might not scale well for real-time/online implementation.
- Because STESSs consider a much larger horizon, the level of uncertainty that they need to consider is also larger. This translates to requiring different methods for modeling uncertainty as the existing methods for building bidding functions are based on scenario generation and stochastic programming. In this context, while there are several methods to generate time-correlated scenarios for short-term horizons (Pinson et al. (2009)), creating time-correlated scenarios with hourly resolutions and large horizons remains, to the best of our knowledge, an open problem.

To tackle these problems, in this paper we propose a control algorithm to build bidding functions and trade electricity that is tailored to systems with long optimization horizons, such as STESSs. The proposed algorithm has two novel features that improve upon the existing methods:

(1) Using *reinforcement learning (RL)*, the algorithm removes the necessity to generate scenarios of price and demand for long-term horizons.
(2) The method scales well for real-time operation independently of the number of variables and/or the size of the horizon. Particularly, as the estimation part of the algorithm is done offline, the time required to build bidding functions is negligible.

These two features are in turn the main contributions of the paper: 1) a control algorithm that generates bidding functions that are tailored to STESSs; 2) an algorithm to build bidding functions that scales well for real-time operation (independently of the characteristics of the underlying system).

To assess the proposed control algorithm, we evaluate its performance using a real STESS, i.e. the Ecovat vessel Ecovat (2018), in a case study where the STESS needs to satisfy an uncertain heat demand during a full year while minimizing the cost of charging the system through the day-ahead market. To assess the performance of the method, we compare it with the theoretical limit that is given by solving the equivalent optimization problem in hindsight, i.e. the solution given by obtaining the optimal cost assuming full knowledge of the prices and demand. The results indicate that, for different yearly-average heat demand levels, the proposed algorithm performs close to this theoretical limit.

The remaining of the paper is organized as follows: Section 2 defines the framework of a generic STESS interacting with the electricity market. Section 3 presents the proposed RL algorithm for building bidding functions. Section 4.1 introduces the STESS considered for the case study. Finally, Section 4 evaluates the algorithm and discusses the results.

2. STESS FRAMEWORK

Before describing the proposed algorithm, we need to define the different properties/variables of a generic STESS when interacting with the electricity market.

2.1 STESS

Let us define the STESS at time $t \in \mathbb{R}$ as a generic dynamical system with an internal state $\mathbf{x}(t) \in \mathbb{R}^{n_x}$; in the case of a STESS, $\mathbf{x}(t)$ usually represents the state of charge of the system. Let us then define its control inputs by $\dot{\mathbf{Q}}^{\text{in}}(t) \in \mathbb{R}^{n_{\text{in}}}$ and $\dot{\mathbf{Q}}^{\text{out}}(t) \in \mathbb{R}^{n_{\text{out}}}$, which respectively represent the rate of energy that is inputted and outputted into/from the system. Finally, let us assume that the system is disturbed by some uncontrollable input $\mathbf{d}(t) \in \mathbb{R}^{n_{\text{d}}}$, e.g. the external temperature.

Then, let us define the uncertain energy demand that the STESS has to supply as $\dot{Q}^{\text{d}}(t)$. Similarly, based on the market bids, let us define the power that the STESS gets allocated from the market as $\dot{Q}^{\text{m}}(t)$. Assuming that the STESS uses the stored energy to supply the heat demand and the allocated power to charge the STESS, the following holds:

$$\dot{Q}^{\text{d}}(t) = \sum_{i=1}^{n_{\text{out}}} \dot{Q}_i^{\text{out}}(t). \tag{1}$$

$$\dot{Q}^{\text{m}}(t) = \sum_{i=1}^{n_{\text{in}}} \dot{Q}_i^{\text{in}}(t). \tag{2}$$

Note that the definition above is very generic and should be able to fit any STESS framework. In particular, as the state and input dimensions are not restricted, this definition includes the following classical STESSs:

- A system composed of independent storage units, e.g. small (distributed) units comprising a large STESS (our case study).

- A system that has different charging/discharging devices with different prices/efficiencies, e.g. heat pumps, electric boilers, etc.

2.2 Optimization Problem

Given a periodic seasonal cycle of N days, an unknown heat demand $\dot{Q}^{\mathrm{d}}(t)$, and a day-ahead market with unknown daily hourly prices[1] $[\lambda_1, \ldots, \ldots \lambda_{24}]^\top$, the goal of the control algorithm is to build hourly optimal bidding curves $\dot{Q}^{\mathrm{b}}_h(\lambda)$ during the periodic seasonal cycle.

In particular, the controller should build one day in advance 24 optimal bidding curves $\dot{Q}^{\mathrm{b}}_1(\lambda), \ldots, \dot{Q}^{\mathrm{b}}_{24}(\lambda)$ such that, while the STESS has always enough energy to satisfy the demand $\dot{Q}^{\mathrm{d}}(t)$, the cost of the purchased power $\dot{Q}^{\mathrm{m}}(t)$ is minimized. Note that, in this market structure, the purchased power $\dot{Q}^{\mathrm{m}}(t)$ at every hour h is defined by:

$$\dot{Q}^{\mathrm{m}}(t) = \dot{Q}^{\mathrm{b}}_h(\lambda_h), \quad \forall\, t \in [h, h+1], \tag{3}$$

with λ_h being the market cleared price during hour h.

3. CONTROL ALGORITHM

To solve the described problem, i.e. how to build bidding functions for STESS, in this paper we propose a RL algorithm based on fitted Q-iteration (Ernst et al. (2005)).

3.1 RL introduction

As any RL algorithm, the proposed method considers that the STESS and its interaction with the environment can be modeled via a Markov decision process (Ernst et al. (2005); Sutton and Barto (2018)). In detail, the system is modeled by a state $\mathbf{s}$, is controlled by an agent that takes actions $\mathbf{a}$ among a discrete set of actions $\mathbb{A} = \{\mathbf{a}^1, \ldots \mathbf{a}^M\}$, and lives in a discrete-time world. In addition, at every discrete time step k, the agent takes an action $\mathbf{a}_k$ and transitions from state $\mathbf{s}_k$ to $\mathbf{s}_{k+1}$ based on some probabilistic dynamics $p(\mathbf{s}_{k+1}|\mathbf{s}_k, \mathbf{a}_k)$. In the transition, the agent receives a reward r_k based on a distribution $q(r_k|\mathbf{s}_k, \mathbf{a}_k)$.

During the training, the RL agent aims at learning an optimal policy $\pi^\star(\mathbf{s}_k)$ that outputs, for each state $\mathbf{s}_k$, the optimal action $\mathbf{a}^\star_k$ so that the expected value of the cumulative sum of discounted rewards R is maximized:

$$R = \sum_{k=1}^{T} \gamma^{T-k} \underset{q(r_k|\mathbf{s}_k,\mathbf{a}_k)}{\mathbb{E}} \{r_k\}. \tag{4}$$

In the expression above, T is the length of a RL episode, i.e. it indicated for how long the RL agent takes decisions, and γ is a discount factor that prioritizes earlier rewards and allows R to be finite even for infinite horizons.

3.2 State and control spaces

For the proposed method, the state $\mathbf{s} = (\mathbf{x}, z, \lambda)$ is defined by three different features:

(1) The current state $\mathbf{x}$ of the STESS, i.e. usually the state of charge of the system.
(2) The current time position z within the periodic seasonal cycle, e.g. for a periodic seasonal cycle of a year z could be the day of the year.
(3) The market price λ.

The reason for selecting these three features is twofold:

- By including them, we can base the selection of the optimal action $\mathbf{a}^\star = \pi^\star(\mathbf{s})$ on both the state of the STESS (given by $\mathbf{x}$) and the state of the environment (given by z and λ).
- Since $\pi^\star(\mathbf{s}) = \pi^\star(\mathbf{x}, z, \lambda)$, given a fixed time point $\hat{z}$ and STESS state $\hat{\mathbf{x}}$, we have a function $\mathbf{a}^\star = \pi^\star(\hat{\mathbf{x}}, \hat{z}, \lambda) = \hat{\pi}^\star(\lambda)$ that selects optimal actions based only on prices. Then, by simply including as a component of the action vector $\mathbf{a}$ the power to be purchased from the market, the bidding function $\dot{Q}^{\mathrm{b}}(\lambda)$ is directly defined by the optimal policy $\hat{\pi}^\star(\lambda)$.

To define the action space $\mathbb{A}$, we consider that a single action $a \in \mathbb{R}^{n_{\mathrm{in}}+1}$ has the following format:

$$a = (\dot{Q}^{\mathrm{in}}_1, \dot{Q}^{\mathrm{in}}_2, \ldots, \dot{Q}^{\mathrm{in}}_{n_{\mathrm{in}}}, j), \quad \forall\, j = 1, \ldots n_{\mathrm{out}}. \tag{5}$$

In particular, we consider that each input control $\dot{Q}^{\mathrm{in}}_i$ ($\forall\, i = 1, \ldots n_{\mathrm{in}}$) can take $D+1$ discrete values defined in a uniform grid between 0 and the maximum power per input control $\dot{Q}^{\max}_i$. In addition, we consider that the control action for the output is given by the selection of an integer j that defines which output $\dot{Q}^{\mathrm{out}}_j$ ($\forall\, j = 1, \ldots n_{\mathrm{out}}$) provides the demand $\dot{Q}^{\mathrm{d}}$. Finally, the action space is defined by all the possible combinations of these values:

$$\begin{aligned}\mathbb{A} = \big\{&(0, \ldots, 0, 1), (\tfrac{1}{D}\dot{Q}^{\max}_1, 0, \ldots, 0, 1),\\ &(\tfrac{2}{D}\dot{Q}^{\max}_1, 0, \ldots, 0, 1), \ldots, (\dot{Q}^{\max}_1, \dot{Q}^{\max}_2, \ldots, \dot{Q}^{\max}_{n_{\mathrm{in}}}, 1),\\ &(0, \ldots, 0, 2), \ldots, (\dot{Q}^{\max}_1, \dot{Q}^{\max}_2, \ldots, \tfrac{D-1}{D}\dot{Q}^{\max}_{n_{\mathrm{in}}}, n_{\mathrm{out}}),\\ &(\dot{Q}^{\max}_1, \dot{Q}^{\max}_2, \ldots, \dot{Q}^{\max}_{n_{\mathrm{in}}}, n_{\mathrm{out}})\big\}.\end{aligned} \tag{6}$$

3.3 Reward function

As the goal of the algorithm is to minimize the cost of purchasing energy, the reward function r_k at time step k is defined as the negative of the cost of the energy purchased at time step k. In particular, if the system is at state $\mathbf{s}_k = (\mathbf{x}_k, z_k, \lambda_k)$ and takes an action $\mathbf{a}_k = (a_{1,k}, \ldots, a_{n_{\mathrm{in}},k})$, the reward is defined as $-\lambda_k \sum_{i=1}^{n_{\mathrm{in}}} a_{i,k}$. In addition, to avoid not having enough energy stored to satisfy the requested demand $\dot{Q}^{\mathrm{d}}_k$, the cost of this situation is set to 10 times the cost of instantaneously buying $\dot{Q}^{\mathrm{d}}_k$ in the market[2]. Finally, as with standard RL algorithms, the reward at the last point in an episode, i.e. $k = T$, is 0. Using these three definitions, the full expression for the reward is then given by:

$$r_k = \begin{cases} 0, & \text{if } k = T\\ -\lambda_k \left(\displaystyle\sum_{i=1}^{n_{\mathrm{in}}} a_{i,k} + 10\, \dot{Q}^{\mathrm{d}}_k\right), & \begin{array}{l}\text{if not enough}\\ \text{energy in STESS}\end{array}\\ -\lambda_k \displaystyle\sum_{i=1}^{n_{\mathrm{in}}} a_{i,k}, & \text{otherwise.}\end{cases} \tag{7}$$

[1] While the algorithm works for any day-ahead market structure, for simplicity we assume that the day-ahead market has hourly prices.

[2] Selecting a factor of 10 is a design choice. The agent just needs a large penalty cost whenever it depletes the STESS.

3.4 Episode length and time grid

The episode length T is defined as two seasonal periodic cycles. The reason for selecting T spanning more than one seasonal cycle is to avoid optimal policies that deplete the STESS at the end of the cycle. In particular, since the state $\mathbf{s}$ considers the time position z within a seasonal cycle, if the episode length is equal to the cycle length, the agent would know its time position within an episode and could use it to deplete the STESS at the end of the episode/cycle. This behavior would be unacceptable as the agent controls the STESS for an unknown number of periodical cycles and depleting the system after one cycle would prevent the agent from providing any energy after this first cycle.

For the size of the discrete time grid, we consider that a time transition $k \rightarrow k+1$ spans a day. The reasons for this are twofold:

- As the system is designed for seasonal storage, it is assumed that its energy content does not change dramatically from one day to another. As such, the optimal bidding curves within a day should be very similar.
- As the decision making process occurs once per day, i.e. we submit 24 bidding functions to the day-ahead market at the same time point, it is convenient to consider time steps of a day.

Note that selecting this time step size is just a design choice. In particular, it is equally possible to consider time steps of one hour and have different bidding functions for each hour of the day. This would however increase the computation load as, for the same seasonal cycle, the length of an episode would increase twentyfourfold.

3.5 Simulation environment

To train the RL agent, we build a simulation environment that recreates the world a STESS lives in. This environment consists of two modules: 1) one that creates realistic price/demand time series that span two seasonal periodic cycles (note that an episode length is two periodic cycles); 2) one that simulates the STESS dynamics.

For the first module, defining by n_h the number of historical data points, we use the dataset $\mathbb{D} = \bigl\{(\lambda_i, \dot{Q}^\mathrm{d}_i)\bigr\}_{i=1}^{n_\mathrm{h}}$ of historical prices and demand. Particularly, to create the price/demand time series for an episode, we randomly pick two consecutive seasonal cycles from $\mathbb{D}$. Then, to add variability to the time series and to create a more robust RL agent, instead of using the historical data point $(\lambda_k, \dot{Q}^\mathrm{d}_k)$ for time step k, we randomly select a point from the dataset $\bigl\{(\lambda_i, \dot{Q}^\mathrm{d}_i)\bigr\}_{i=k-n_\mathrm{w}}^{k+n_\mathrm{w}}$, where n_w is the window size. The selection of n_w is a design choice: for the presented case study, as the seasonal periodic cycle is one year, we select n_w equal to 2 weeks.

For the second module, we simply use the dynamical model of the STESS that describes the evolution of the internal state $\mathbf{x}(t)$ as a function of the two system inputs $\dot{\mathbf{Q}}^\mathrm{in}(t)$ and $\dot{\mathbf{Q}}^\mathrm{out}(t)$ (Lago et al. (2019)).

3.6 Training algorithm

To train the RL agent, we use the fitted Q-iteration algorithm (Ernst et al. (2005)) with boosting trees (Chen and Guestrin (2016)). The reason for selecting this algorithm is twofold: 1) it is a very general algorithm that has been successfully used in multiple applications; 2) we empirically observed that this algorithm performed as good as more advanced RL algorithms, e.g. DQN, but without the additional computational complexity.

In this RL scheme, the algorithm is trained by iteratively performing two steps. In the first step, the RL agent interacts with the environment during a full episode and stores the dataset $\{(\mathbf{s}_k, \mathbf{a}_k, \mathbf{s}_{k+1}, r_k)\}_{k=1}^{T-1}$ of transitions, actions, and rewards that it experiences. To take the action $\mathbf{a}_k$, it uses an ϵ-greedy approach: it takes a random action with a probability ϵ and the optimal action (given by last estimated optimal policy $\bar{\pi}^\star(\mathbf{s}_k)$) with a probability $1-\epsilon$, where ϵ decays after each episode. To make the agent more robust, the initial state $\mathbf{s}_0$ of the episode is randomly selected[3].

In the second step, the algorithm uses the dataset of stored data to estimate an optimal policy $\bar{\pi}^\star(\mathbf{s}_k)$. In particular, it estimates a model $\bar{Q}(\mathbf{s}_k, \mathbf{a}_k)$ of the so-called Q-functions $Q(\mathbf{s}_k, \mathbf{a}_k)$, which represent the expected cumulative reward at state $\mathbf{s}_k$ when taking action $\mathbf{a}_k$ and then acting optimally. Then, an estimation of the optimal policy is built as $\mathbf{a}^\star_k = \bar{\pi}^\star(\mathbf{s}_k) = \underset{\mathbf{a}_k}{\operatorname{argmin}}\, \bar{Q}(\mathbf{s}_k, \mathbf{a}_k)$

These two steps run one after the other until the algorithm converges. We refer to Ernst et al. (2005) and Sutton and Barto (2018) for further details.

3.7 Building bidding functions

After the RL agent is trained, the optimal bidding functions $\dot{Q}^\mathrm{b}(\lambda)$ for any state $\mathbf{x}$ and time z are directly obtained. In particular, since $\pi^\star(\mathbf{s}) = \pi^\star(\mathbf{x}, z, \lambda)$, given a fixed time point $\hat{z}$ and STESS state $\hat{\mathbf{x}}$, we have a function $\mathbf{a}^\star = \pi^\star(\hat{\mathbf{x}}, \hat{z}, \lambda) = \hat{\pi}^\star(\lambda)$. As the action $\mathbf{a}$ contains the power purchased from the market, the bidding function $\dot{Q}^\mathrm{b}(\lambda)$ is directly defined by the optimal policy $\hat{\pi}^\star(\lambda)$.

4. CASE STUDY

To illustrate the performance of the algorithm, we consider a thermal stratified storage tank, the Ecovat vessel Ecovat (2018), in a case study where the STESS needs to satisfy an uncertain heat demand during a full year while minimizing the cost of charging the system through the day-ahead market.

4.1 Real STESS

The considered thermal vessel is a large subterranean thermal storage vessel with capabilities for seasonal thermal storage and with the ability to supply heat demand to a cluster of buildings. It is divided into several segments that can be charged and discharged separately. Due to this

[3] Only the $\mathbf{x}_0$ component is actually randomly selected as z_0 is predefined and λ_0 is given by historical data.

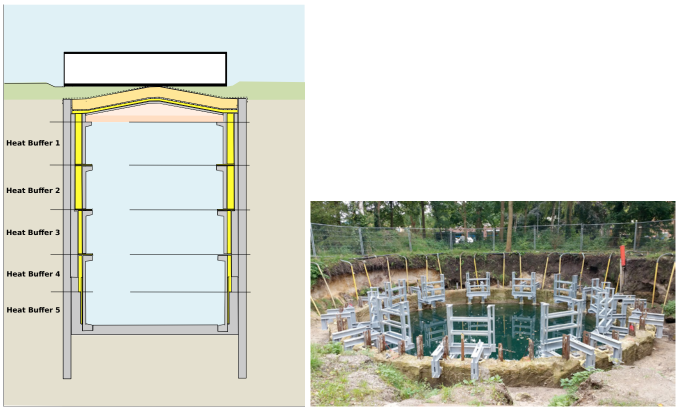

Fig. 1. Left: schematic representation of the STESS. Right: construction of the STESS

property and thermal stratification, each of these segments acts as a different heat buffer. In detail, the system has 5 thermal buffers with the top 4 buffers (see Figure 1) being able to be charged and discharged independently. The system considers a seasonal periodic cycle of a year, i.e. it tries to use the cheap electricity prices of summer to provide heat on the winter.

The insulation structure of the vessel is such that it can very efficiently store energy between seasons: the heat losses of the vessel are about 25% over a period of 6 months. Figure 1 provides a schematic overview of the vessel and the real system when it was under construction. For further details on the system we refer to (Lago et al. (2019)).

4.2 Model and definitions

To model the system, we consider the following definitions:

- The state $\mathbf{x}_k = [T_{1,k}, T_{2,k}, T_{3,k}, T_{4,k}, T_{5,k}]^\top$ at time step k is given by the temperature stored in each of the 5 buffers. We use the temperature in each heat buffer as it is proportional to the stored energy.
- As the top 4 buffers can be charged and discharged independently, we respectively define $\dot{\mathbf{Q}}_k^{\text{in}} = [\dot{Q}_{1,k}^{\text{in}}, \ldots, \dot{Q}_{4,k}^{\text{in}}]^\top$ and $\dot{\mathbf{Q}}_k^{\text{out}} = [\dot{Q}_{1,k}^{\text{out}}, \ldots, \dot{Q}_{4,k}^{\text{out}}]^\top$ as the input and output power at time step k.
- For the position z within the seasonal periodic cycle, we use the day of the year, i.e. $z \in \{1, \ldots 365\}$.

For the STESS simulation model required for the RL environment, we employ the dynamical model for thermally stratified storage tanks defined by Lago et al. (2019).

4.3 Data

To evaluate the algorithm, we consider three years of historical data involving the day-ahead prices of 2015, 2016, and 2017 in the Dutch day-ahead market [4], and the heat demand of a cluster of 5 buildings with a yearly-average heat demand of 220 MWh during the same time period [5].

[4] Collected from `https://transparency.entsoe.eu/`.
[5] Obtained from one of our research partners.

4.4 Experimental Setup

The RL agent is first trained using the data from 2015 and 2016. Next, the agent is evaluated on the 2017 dataset. In particular, at hourly steps, we discharge the STESS following the demand of 2017. In parallel, we use the agent to build the optimal bidding functions at every day and we employ the prices of 2017 as the market clearing prices.

For the evaluation, we compute the economic savings that the RL agent provides w.r.t. not having a heat buffer and directly buying the instantaneous heat demand $\dot{Q}^{\text{d}}$ at the day-ahead market price. Then, we compare the savings of the RL agent w.r.t. the maximum theoretical savings. This theoretical upper bound is obtained by solving the optimization problem that minimizes the cost in hindsight assuming perfect knowledge of the demand and price (Lago et al. (2019)). Note that, due to the uncertainty in prices and demand, this theoretical limit is impossible to reach in practice.

Finally, to evaluate the robustness of the algorithm and to compare its relative performance under different conditions, the RL agent is evaluated at 20 different random initial states $\mathbf{x}_0$. In addition, the demand data is multiplied by 2 and used to retrain and evaluate the algorithm in the case of having 10 buildings, i.e. having a yearly-average demand of 440 MWh.

4.5 Results

The results of training the RL agent are depicted in Figure 2 representing, for the 220 MWh scenario, the evolution of the yearly cost (average cost over the two seasonal cycles comprised in an episode) as a function of the number of episodes explored. As it can be seen, the RL agent converges to a solution after 200-300 episodes; while this does not mean that the solution is good, it is the first step to ensure that the RL agent is correctly trained.

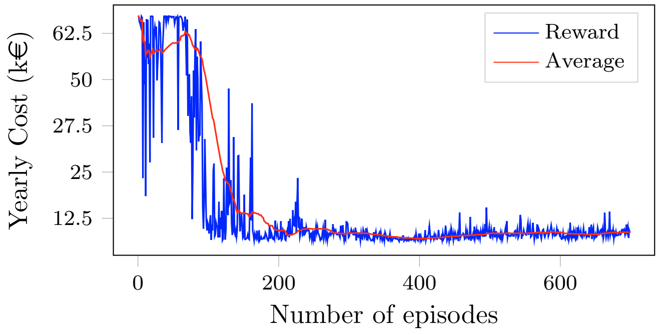

Fig. 2. Training results for the RL agent on a 220 MWh yearly-average demand level.

Table 1 presents the comparison between the RL agent and the theoretical limit. From the table, several observations can be made:

- The RL agent learns to trade energy as the cost of the RL agent is lower than the cost of directly buying the energy, i.e. not having an STESS.
- The RL agent is robust and leads to stable solutions: despite considering 20 different random initial states

$\mathbf{x}_0$ and two different demand levels, the standard deviation of the solutions is just 2-3 %.

- In terms of the quality of the solutions, considering the uncertainty and volatility of day-ahead prices, the RL agent performs reasonably well: while in hindsight (theoretical limit) we could respectively save 24 % and 31 % for the 440 MWh and 220 MWh case, the RL agent saves approximately 12 % and 22 %.

Table 1. Comparison in terms of economic cost and savings during 2017 between the RL agent and the theoretical limit. The costs are computed as the average costs for 20 random initial points $\mathbf{x}_0$. The savings are computed w.r.t. the cost of not having a STESS.

		Average demand	
		440 MWh	**220 MWh**
Cost [€]	**No STESS**	15005.5	7502.8
	Min. theoretical	11331.3	5151.8
	RL Agent	13264 ± 2 %	5876.9 ± 3 %
Savings	**Max. theoretical**	24.1%	31.3%
	RL Agent	11.6%	21.6%

4.6 Discussion

Based on the obtained results, it can be stated that the RL agent learns how to trade energy and to build bidding functions that improve the cost of providing heat demand via a STESS. Moreover, not only does the algorithm build bidding functions that reduce the cost, but it is also very robust: for different initial points and for different demand levels the solutions are of the same quality.

Two of the advantages of the algorithm are that—when compared with the approaches proposed in the literature for traditional retailers—it removes the necessity for creating future scenarios and it can be used online independently of the number of variables or the size of the horizon.

In detail, as the RL agent is trained based on historical data, generating future scenarios of the price and demand is not necessary. Similarly, as all the training is done offline, the online computational cost of the bidding functions is negligible and independent of the number of variables. These two features are key for STESS systems as: 1) generating realistic scenarios for very long-term horizons is, to the best of our knowledge, an open problem; particularly, due to accumulation of errors, generating realistic long-term scenarios is extremely difficult. 2) The computational cost of solving online optimization problems for long-term horizons can quickly become larger than the time that the STESS has to make a bid, i.e. some hours/minutes.

In terms of the quality of the solution, the RL agent obtains financial costs that are between the two extreme scenarios, i.e. perfect knowledge of the future and not having a STESS system. Considering the large uncertainty and volatility of electricity prices over the course of a year, i.e. how utopian and unreal is the case of having perfect knowledge of the future, the obtained results are arguably good. In particular, for different demand scenarios and initial points, the RL agent is able to obtain robust solutions that provide financial savings that are equal to 50-70 % of the maximum theoretical limit.

5. CONCLUSION

In this paper a novel approach for generating bidding functions for seasonal thermal energy storage systems (STESSs) has been proposed. The algorithm is based on *reinforcement learning (RL)*, and it has been shown to provide robust trading strategies and financial savings that are relatively close to the maximum theoretical limit. In future research, this work will be expanded to the case of multiple markets and/or different types of devices interacting with the STESS.

ACKNOWLEDGEMENTS

This research has received funding from the European Unions Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No 675318 (INCITE).

REFERENCES

Miguel Carrion, Antonio J. Conejo, and José M. Arroyo. Forward contracting and selling price determination for a retailer. *IEEE Transactions on Power Systems*, 22(4): 2105–2114, 2007.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 785–794, 2016.

Ecovat. Ecovat renewable energy technologies BV, 2018. `http://www.ecovat.eu`.

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

Stein-Erik Fleten and Erling Pettersen. Constructing bidding curves for a price-taking retailer in the Norwegian electricity market. *IEEE Transactions on Power Systems*, 20(2):701–708, 2005.

Li Gong, Shi Jing, and Xiuli Qu. Modeling methods for GenCo bidding strategy optimization in the liberalized electricity spot market–A state-of-the-art review. *Energy*, 36:4686–4700, 2011.

Rocio Herranz, Antonio Muñoz San Roque, Jose Villar, and Fco. Alberto Campos. Optimal demand-side bidding strategies in electricity spot markets. *IEEE Transactions on Power Systems*, 27(3):1204–1213, 2012.

Jesus Lago, Karel De Brabandere, Fjo De Ridder, and Bart De Schutter. Short-term forecasting of solar irradiance without local telemetry: A generalized model using satellite data. *Solar Energy*, 173:566–577, 2018.

Jesus Lago, Fjo De Ridder, Wiet Mazairac, and Bart De Schutter. A 1-dimensional continuous and smooth model for thermally stratified storage tanks including mixing and buoyancy. *Applied Energy (In Press)*, 2019.

Pierre Pinson, Henrik Madsen, Henrik A. Nielsen, George Papaefthymiou, and Bernd Klöckl. From probabilistic forecasts to statistical scenarios of short-term wind power production. *Wind Energy*, 12:51–62, 2009.

Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: An introduction.* MIT press, 2018.

Jing Xu, Ruzhu Wang, and Yong Li. A review of available technologies for seasonal thermal energy storage. *Solar Energy*, 103:610–638, 2014.