Optimal Energy Scheduling of Flexible Industrial Prosumers via Reinforcement Learning

Nick van den Bovenkamp*‡, Juan S. Giraldo†, Edgar Mauricio Salazar Duque§, Pedro P. Vergara*, Charalambos Konstantinou¶, and Peter Palensky*

* Intelligent Electrical Power Grids – EEMCS, Delft University of Technology, Netherlands

† Energy Transition Studies Group, Netherlands Organisation for Applied Scientific Research (TNO), Netherlands

§ Electrical Energy Systems – EES, Eindhoven University of Technology, Netherlands

‡ Innovation Department, Sunrock Investments B.V., Netherlands

¶ CEMSE Division, King Abdullah University of Science and Technology (KAUST), Saudi Arabia

***Abstract*—This paper introduces an energy management system (EMS) aiming to minimize electricity operating costs using reinforcement learning (RL) with a linear function approximation. The proposed EMS uses a Q-learning with tile coding (QLTC) algorithm and is compared to a deterministic mixed-integer linear programming (MILP) with perfect forecast information. The comparison is performed using a case study on an industrial manufacturing company in the Netherlands, considering measured electricity consumption, PV generation, and wholesale electricity prices during one week of operation. The results show that the proposed EMS can adjust the prosumer's power consumption considering favorable prices. The electricity costs obtained using the QLTC algorithm are 99% close to those obtained with the MILP model. Furthermore, the results demonstrate that the QLTC model can generalize on previously learned control policies even in the case of missing data and can deploy actions 80% near to the MILP's optimal solution.**

***Index Terms*—Q-learning, tile coding, energy management system, mixed-integer linear programming**

I. Introduction

Buildings incorporating a battery energy storage system (BESS), and photovoltaic (PV) systems, are expected to play a critical role in future power systems [1]. The growing share of flexible resources (e.g., BESS, electric vehicles) and renewable energy sources (e.g. PV), as well as the introduction of new local market mechanisms, enables the transformation of classic consumers into prosumers. Due to their features, prosumers are attractive candidates to participate in demand response (DR) programs, e.g., through time-varying prices, balancing service offerings, or congestion relieving [2]. To do this, prosumers must rely on an Energy Management System (EMS), which aims to define the optimal control strategies by optimizing the planning and operation of the various energy resources, aiming for a specified objective (usually minimizing total operational cost) [3]. Mathematical optimization formulations based on mixed-integer linear programming (MILP) and stochastic programming can define an optimal energy management strategy [4]. However, their computational burden increases rapidly if the problem grows in complexity, e.g., more sophisticated system dynamics or more decision variables are added. Another disadvantage of such classical methods is the required time for updating a control strategy since reacting in time can become infeasible.

Reinforcement learning (RL) algorithms offer an alternative solution to the drawbacks faced by classic mathematical formulations. RL algorithms learn a control strategy by directly interacting with a dynamic system (its environment). By doing so, an RL agent assesses the performance of a sequential decision for each possible state-action pair by a reward function. Conventional Q-learning calculates a Q-value and recursively stores it in a table. If the size of this table is large enough and the RL algorithm is well designed, the obtained policy function can generalize the agent's optimal operation [5]. However, storing these values becomes infeasible as the number of state-action pairs increases, for instance, when considering continuous variables. This problem is known in the literature as the *curse of dimensionality*. A solution for this problem is approximating the Q-values using parametric or non-parametric functions [6], [7].

Recent showed that deep reinforcement learning (DRL) algorithms can achieve satisfactory control strategies for energy management problems [8]. For instance, the authors in [9] proposed a DRL to control a battery in a small microgrid using deep Q-networks (DQNs). Similarly, a double deep Q-learning algorithm is introduced in [10]. However, since DRL methods are based on artificial neural networks, global convergence is not guaranteed and might present stability issues due to hyperparameter settings [11]. Therefore, DRL algorithms need extensive hyperparameter tuning to achieve stable performance [12]. Linear function approximation methods can also effectively approximate Q-values [6], [13]. Compared to DRL, linear function approximation methods might showcase less generalization potential yet better convergence guarantees than DQNs [14]. Convergence guarantees should be prioritized overbroad generalization since applicable generalization is only useful with convergence to good-quality solutions [15]. Another advantage is that their linear behavior is far more transparent than these non-linear function approximation methods, making these methods more favorable from a debugging and engineering point of view. Although several RL algorithms have been used for the optimal energy scheduling problem [16], to the best of the authors' knowledge, Q-learning with tile coding (QLTC) has yet to be used for flexible industrial prosumers. Therefore, the main contribution of this paper is to introduce an EMS for a flexible industrial prosumer based on Q-learning with tile coding. The proposed algorithm can learn an effective control policy by minimizing the day-ahead electricity costs. The proposed approach also includes a reward function for fast convergence.

II. RL and Tile Coding Background

In RL, a decision-making problem is expressed in the form of a finite Markov Decision Process (MDP), defined as the tuple $\langle \mathcal{S}, \mathcal{A}(s), \mathcal{R}, p, \gamma \rangle$, where $\mathcal{S}$ is the *state space*, $\mathcal{A}(s)$ the *action space*, $\mathcal{R}$ the *set of rewards*, which are all finite, and $\gamma \in$ (0,1] is the *discount factor*. Variable $p : \mathcal{S} \times \mathcal{S} \times \mathcal{A}(s) \rightarrow$ [0,1] is the state-transition probability function and describes the environment's dynamics, usually not available and must be learnt by interaction. The RL agent's goal is to maximize the cumulative discounted reward, i.e., the *return* G_t, following:

$$G_t \doteq \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{1}$$

A RL agent follows a *policy* defined as a mapping function that determines the action selection from a certain state, denoted with $\pi(s) : \mathcal{S} \rightarrow \mathcal{A}(s)$. The *action-value function* or *Q-value function* estimates how good a particular action is in a given state while following policy π by estimating the expected return, E_π.

$$Q_\pi(s,a) \doteq \mathbb{E}_\pi \left[G_t \middle| S_t = s, A_t = a \right] \tag{2}$$

Eventually, in the case of a finite MDP, the agent finds an optimal policy π_* that yields the highest reward. The optimal Bellman equation that describes the relationship between consecutive Q-values is then given by:

$$\begin{aligned} Q_*(s,a) &= \mathbb{E}\left[R_{t+1} + \gamma \max_{\pi} Q_*(S_{t+1}, a') \middle| S_t = s, A_t = a \right] \\ &= \sum_{s',r} p(s', r|s,a) \left[r + \gamma \max_{a} Q_*(s', a') \right] \end{aligned} \tag{3}$$

Q-learning directly approximates the action-value function by updating estimates based on other estimates, i.e., bootstrapping. Hereby, it has relatively fast learning capabilities. A Q-function, when approximated by a linear function, can be expressed as:

$$\hat{Q}(s,a,\mathbf{w}) \doteq \mathbf{w}^\top \mathbf{x}(s,a) \doteq \sum_{i=1}^{d} w_i x_i(s), \tag{4}$$

where $\mathbf{w}$ is a parameter vector, d the dimensionality, and $\mathbf{x}(s,a)$ the feature vector that represents state-action pairs. Tile coding linear function approximation divides the entire state space into smaller grid-like sub-parts. Each sub-part is called a tile, m, and has a component m_i for each state variable, $|m|$ = $|\mathcal{S}|$. One layer of tiles together is called a tiling, n. Multiple layers of tilings are present in tile coding, each with a small offset of $1/n$. The number of tiles and tilings together determine the function approximation resolution in the corresponding state space dimension, described by $1/(m_i \cdot n)$. Fig. 1 shows the configuration of two different tile coding settings

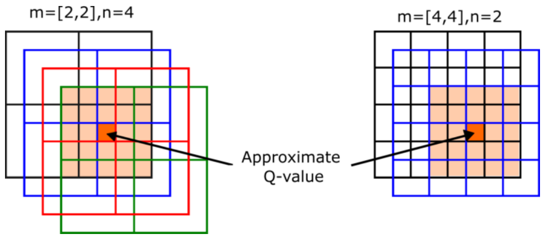
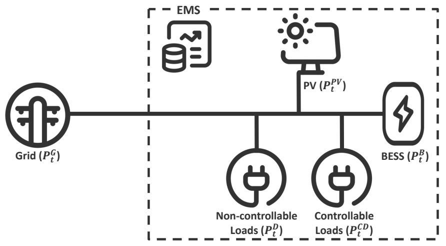


Fig. 1: The Q-value function approximation for a two-dimensional state space under different tile coding settings and an identical resolution.


EMS
PV (P_t^{PV})
Grid (P_t^G)
BESS (P_t^B)
Non-controllable Loads (P_t^D)
Controllable Loads (P_t^{CD})
Fig. 2: The energy management system's layout and its components.

that result in an identical Q-function approximation resolution. The binary feature vectors, used to represent the feature vector $\mathbf{x}(s,a)$, make tile coding computationally very efficient, since the number of active features is constant for every state. Instead of performing d amount of additions and multiplication, it adds up the n amount of active parameters, making tile coding the most functional feature representation for modern computers. For a more detailed explanation, see [15].

III. System Layout and Boundary Conditions

As illustrated in Fig. 2, the smart building's energy management problem consists of an electricity consumer with controllable and non-controllable loads, local PV generation P_t^{PV}, and a BESS P_t^B. All controllable and non-controllable loads present in the system aggregate to a controllable demand (CD) and a non-controllable power demand denoted with P_t^{CD} and P_t^D, respectively. The power balance for every discrete time step t is expressed as:

$$P_t^G = P_t^D - P_t^{PV} + P_t^B + P_t^{CD}, \tag{5}$$

where P_t^G denotes the grid power demand, that has a one-way transport limit $P_{min}^G \leq P_t^G$. The smart building participates in an DR program where the EMS receives the wholesale day-ahead prices (DAP), expressed as:

$$\min \ C = \sum_{t \in T} e_t P_t^G \Delta t, \tag{6}$$

where C denotes the electricity costs, e_t the DAP, Δt the difference in hours between two subsequent time steps, and T is the set of discrete time steps. The CD is modeled as a tuple of CD actions, $a^{CD}(s)$, as shown in (7). It is assumed that the CD can always increase or decrease its power output with the same case-specific maximum CD power rate, P_{max}^{CD}.

$$a^{CD}(s) = \left[-P_{max}^{CD}, 0, P_{max}^{CD} \right]. \tag{7}$$

At each time step, the discrete action taken equals the power output of the controllable demand $P_t^{CD} = a_t^{CD}(s)$. The power demand shifted at t is denoted with ΔP_t^{CD} and updates according to the following relationship:

$$\Delta P_{t+1}^{CD} = \Delta P_t^{CD} + P_t^{CD}. \tag{8}$$

The shifted power is limited by one time the maximum CD power, given by $-P_{max}^{CD} \leq \Delta P_t^{CD} \leq P_{max}^{CD}$, ensuring the DR has minimal impact on regular business operation. Another constraint is that the total amount of power shifted over one day is equal to zero, $\Delta P_{|T|}^{CD}$ = 0. Controlling the loads does not impact the overall power consumption.

The SOC of the BESS updates according to:

$$SOC_{t+1} = SOC_t + \frac{P_t^B \Delta t}{E^B}, \tag{9}$$

where E^B denotes the capacity of the BESS. The SOC should stay within in its limits, $SOC_{min} \leq SOC_t \leq SOC_{max}$. The initial and final SOC are denoted with SOC_0 and $SOC_{|T|}$, respectively. Whereas $SOC_{|T|}$ = SOC_0 to guarantee continuous operation between subsequent days. The power difference between each discrete action is defined by ΔP^B = $2P_{max}^B/(X^B - 1)$, for which P_{max}^B is the maximum power output of the BESS. The following relationship describes the entire set of discrete BESS actions:

$$a^B(s) = \big[-P_{max}^B, (\Delta P^B - P_{max}^B), (2\Delta P^B - P_{max}^B), \\ ..., (P_{max}^B - 2\Delta P^B), (P_{max}^B - \Delta P^B), P_{max}^B \big]. \tag{10}$$

The BESS's ramp rate constraint prevents it from fast degradation by limiting the current charge rate by the difference between P_{max}^B and the previous charge rate:

$$P_{t-1}^B - P_{max}^B \leq P_t^B \leq P_{t-1}^B + P_{max}^B. \tag{11}$$

IV. Q-Learning with Tile Coding

This section explains the various elements of the proposed QLTC approach. The proposed QLTC uses the decaying ϵ-greedy technique to balance exploration and exploitation, described by the decaying function $\epsilon = \frac{1}{1+(\lambda\, i)}$, where λ is the decay hyperparameter and i the i-th episode the agent encounters. The pseudocode for the learning process of the QLTC is depicted in Algorithm 1.

A. State Space and Action Space

This study proposes a state space design that minimizes the number of state variables for fast convergence and generalization while retaining enough variables to achieve high accuracy. Constructing $P_t^N = P_t^D - P_t^{PV}$ reduces the number of state variables while still taking both generation and consumption into account. The relative electricity price enhances generalization between different training days. A state at t is given by:

$$s_t = \big\langle t, P_t^N, SOC_t, \Delta P_t^{CD}, (\hat{e}_t - \overline{e}) \big\rangle, \qquad s_t \in \mathcal{S} \tag{12}$$

The action space is a combination of the set of actions from the BESS and the controllable demand, $|\mathcal{A}(s)|$: $a^B(s)$ $\times$ $a^{CD}(s)$, given in (13). Therefore, the size of the state space scales linearly with X^B. The set of actions is state-dependent since it is blinded based on whether the particular state is located at an active boundary condition or not.

$$\mathcal{A}(s) = \big\langle a^B(s), a^{CD}(s) \big\rangle \tag{13}$$

Algorithm 1 Q-learning with tile coding learning process

```
 1: Inputs:
      Hyperparameters:       $\alpha$, $\gamma$, $\lambda$, episodes, $m$, $n$, IHT size
      Variables:             $P_{max}^{CD}$, $P_{max}^B$, $X^B$, $E^B$, $SOC_{min}$,
      $SOC_{max}$, $SOC_0$, $P_{min}^G$
      Data sets:             $e_t$, $P_t^D$, $P_t^{PV}$     $\forall t \in T$
      Q-values (optional):   IHT, w
 2: Initialize:
      Action space
 3: for each episode do
 4:     Initialize:
        Initial state $S_0$
 5:     for each time step do
 6:         Blind: determine $a(s)$
 7:         Take $a_t$ depending on $\epsilon$-greedy
 8:         Receive $r_{t+1}^1$, $r_{t+1}^2$ and $s_{t+1}$
 9:         Blind: determine $a(s)$
10:         Determine $\max_a \hat{Q}(s_{t+1}, a, \mathbf{w})$
11:         $\mathbf{w} \leftarrow \mathbf{w} + \alpha[r_{t+1}^2 + \gamma \max_a \hat{Q}(s_{t+1}, a, \mathbf{w}) - \hat{Q}(s_t, a_t, \mathbf{w})]$
12:         $s_{t+1} \leftarrow S_t$
13:     end for
14: end for
15: Outputs:
      IHT, w
```

B. Reward Function

Since RL agents maximize their return, the objective function is reformulated as maximizing the negative electricity costs C_{QLTC}, resulting in the reward function given by (14). This representation is the standard approach for energy management problems in literature.

$$r_t^1 = -e_t P_t^G \Delta t \tag{14}$$

This study proposes a new reward function design for energy management problems, given in (15). It subtracts the daily average electricity price from the standard function at the corresponding time step.

$$r_t^2 = -(e_t - \overline{e}) P_t^G \Delta t \tag{15}$$

A MILP optimization model has been implemented, mimicking the proposed QLTC. The optimal electricity costs reached by the MILP optimization method are denoted by C_{MILP} and calculated following the expression in (6). Similarly, costs reached by the QLTC algorithm are denoted by C_{QLTC}. The QLTC performance is assessed as a percentage of the relative electricity costs between the QLTC and the MILP. This performance is denoted by η in (16). C_0 is the electricity costs for a zero power output control strategy for the BESS and CD.

$$\eta = \frac{C_{\text{QLTC}} - C_0}{C_{\text{MILP}} - C_0} \cdot 100\% \tag{16}$$

TABLE I: Variables and hyperparameter settings.

CD	P^{CD}_{max} = 50 kW
BESS	X^B = 11, P^B_{max} = 50 kW, E_B = 75 kWh $SOC_{min} = 20\%, SOC_{max} = 100\%, SOC_0 = 20\%$
Grid	P^G_{min} = -560 kW
QLTC	α_0=0.64 , γ = 1.0, 15000 episodes, λ = 0.0006 IHT size = 8388608 m = $\left[\, 24, 10, 10, 3, 20 \,\right]$, n = 32

V. Case study

The effectiveness and performance of the proposed algorithm are evaluated on a smart building in the Netherlands that operates heavy industrial machinery, i.e., heating, ventilation, and air conditioning systems, an electric water heater, and other building-related devices. These continuous processes can be adjusted partly but not stopped entirely. The seasonal power consumption patterns indicate that 50 kW of power demand is controllable. The smart building's 800 kWp PV system can deliver a maximum of 560 kW of power to the grid. The BESS sizes to a maximum power output of 50 kW and a capacity of 75 kWh. The utilized variables and the hyperparameter settings of the QLTC agent are summarized in Table I.

A. Training, Validation, and Test Sets

Consumption, PV generation, and DAP prices data from June-August are examined and divided into a training, validation, and test set. The validation and test sets consist of one week of data starting from Monday, July first and Monday, July 15th, respectively. The training set consists of the three summer months minus the test set. Hyperparameter tuning and operational performance evaluation is done by training and directly deploying the learned policy on the same day as the validation set data. Generalization is evaluated by training the agent on the entire training set and deploying the learned policy on the test set, where the agent learns upon the previously learned Q-values during training.

B. Operational Performance on the Validation Set

The QLTC agent's convergence is proved by learning repetitively on the first day of the validation set without memorizing previously learned Q-values. Fig. 3 (a) shows that the QLTC's moving average (MA) returns with 95% confidence interval (CI) consistently converges to a value close to the negative C_{MILP}. This subfigure proves that the QLTC's control policy convergence is guaranteed, provoked by the tile coding's linear function approximation on a relatively small optimization problem. The greedy policy depicted in Fig. 3 (b) illustrates the policy improvement throughout the learning process. After roughly 5700 episodes, the MA return stabilizes at an identical value as the MILP, demonstrating that the agent repetitively finds an equal rewarding control strategy.

Fig. 4 shows the operation for the entire week of the validation set. Subfigure (b) depicts that the agent operates accordingly, it increases power output during low prices, and it decreases power output during high prices. Subfigure (c) shows that the QLTC's control policy of the BESS is very similar to the MILP control strategy. Between hours 0 & 26, 45 & 60, and 146 & 157 the charge-discharge cycle are identical. Between time steps 31 & 34, the MILP model discharges and charges, while the QLTC gives zero output.

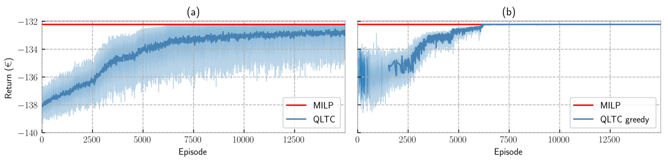


Fig. 3: MA return with 95% CI for 20 repetitive simulations on the first day of the validation set. (a) Complete policy. (b) Greedy policy.

TABLE II: Electricity costs of the QLTC compared to the MILP optimum and a zero output control policy.

	Mon 1st	Tue 2nd	Wed 3rd	Thu 4th	Fri 5th	Sat 6th	Sun 7th	
	0h-24h	24h-48h	48h-72h	72h-96h	96h-120h	120h-144h	144h-168h	**Total**
$C_{\textbf{MILP}}$	132.22	116.69	89.05	96.84	122.50	11.17	7.10	**€575.57**
$C_{\textbf{QLTC}}$	132.22	116.70	89.05	96.91	122.56	11.32	7.11	**€575.85**
C_0	138.22	122.07	95.05	101.26	127.49	14.47	10.26	**€608.82**
η	100%	99.94%	100%	98.46%	98.92%	95.49%	99.78%	**98.94%**

Table II shows that for all strategies, the electricity costs during weekdays are significantly higher than during weekend days, explained by the difference in grid demand. Both control strategies obtain approximately €33 of electricity cost savings for one week of operation compared to C_0. The QLTC agent finds a control policy above 98% near the MILP optimum for each day of the validation set, except for Saturday, July 6th. Nonetheless, this slightly underperforming day reaches an η above 95%. On average, the QLTC reaches a remarkable 98.94% near the MILP's optimum.

C. Generalization on the Training Set

The agent follows the training process earlier described in Section V-A. Fig. 5 (a) shows that for all three training days the MA return consistently converges. Subfigure (b) depicts that the QLTC agent settles at a control policy of η = 99% for all three days. What stands out is that the greedy return already starts at 40% for the third training day, indicating that the agent already encountered similar state-action pairs in previous training.

When a particular range of states is not present in the training set, the agent has not determined accurate approximate Q-values for those states. The relatively low performance of 58.9% on Saturday, July 20th, can be designated to this phenomenon. Fig. 6 (b) shows that the QLTC deploys an unfavorable decision by increasing power output at t= 121, a relatively high price. Also, the significant difference in SOC and ΔP^{CD} cycles between the two models indicate poor approximation. Table III shows that the best performing days are July 16th and 17th reaching an η of 91.5% and 95.7%, respectively. The SOC and ΔP^{CD} cycles appear very similar

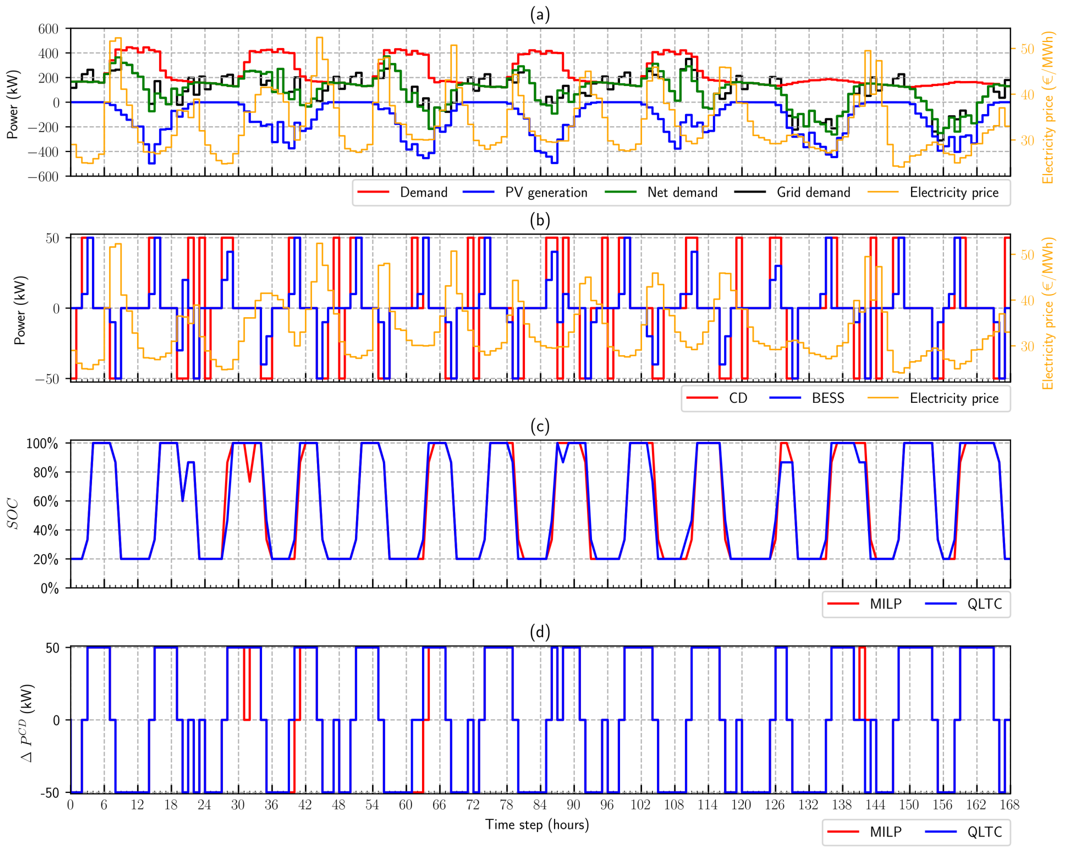


Fig. 4: Operation on the validation set. (a) Power levels, and price. (b) Power BESS, CD, and price. (c) BESS's SOC (d) CD shifted.

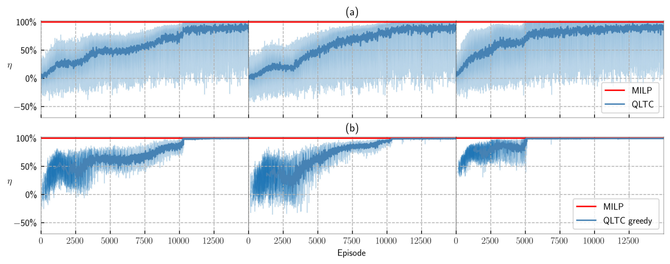


Fig. 5: MA return with 95% CI for five repetitive simulations on the first three days of the test set. (a) Complete policy. (b) greedy policy.

for those two days. On average, the QLTC agent finds a control policy of η = 80.7% on the test set.

A good control policy lowers grid demand during high-price moments and increases grid demand during low-price moments. Due to this strong dependency on the relative price, the training and testing process is repeated for five different relative price tile settings, m_5, without changing the other four tile sizes. The last column in Table III shows that the agent yields an average η of approximately 80% for 15, 20, and 25 m_5 tiles. Thus, the agent outputs a decent control policy on new data for a range of m_5 tile settings, preceding the need for extensive m_5 hyperparameter search.

TABLE III: The η (%) measure on the test set under different tile hyperparameter settings after training on the training set.

η (%)	15th	16th	17th	18th	19th	20th	21st	
m_5	0h	24h	48h	72h	96h	120h	144h	**Avg**
10	70.6	70.7	92.9	25.0	69.4	62.7	65.7	**65.3**
15	74.5	90.6	92.1	91.2	82.3	63.8	71.0	**80.8**
20	86.8	91.5	95.7	79.7	78.4	58.9	74.0	**80.7**
25	86.0	92.1	91.5	84.5	80.5	47.2	74.8	**79.5**
30	86.6	83.7	96.3	73.0	65.1	50.6	76.1	**75.9**

VI. Conclusions

This research developed an EMS that minimizes the electricity costs for a smart building, using RL with linear function approximation. The proposed QLTC EMS has a novel reward function for faster and more stable return convergence. The QLTC also has a clever state space design that minimizes the number of state variables to enhance generalization while maintaining strong convergence. Another feature of the QLTC's design is aggregating controllable loads for better convergence, implementation, and scalability. The QLTC EMS effectively minimized the smart building's electricity costs in a case study by learning and deploying a control policy for the next day of operation 99% near the MILP's optimum. Furthermore, the results showed that the QLTC agent generalizes on previously learned data by deploying a control policy 80%

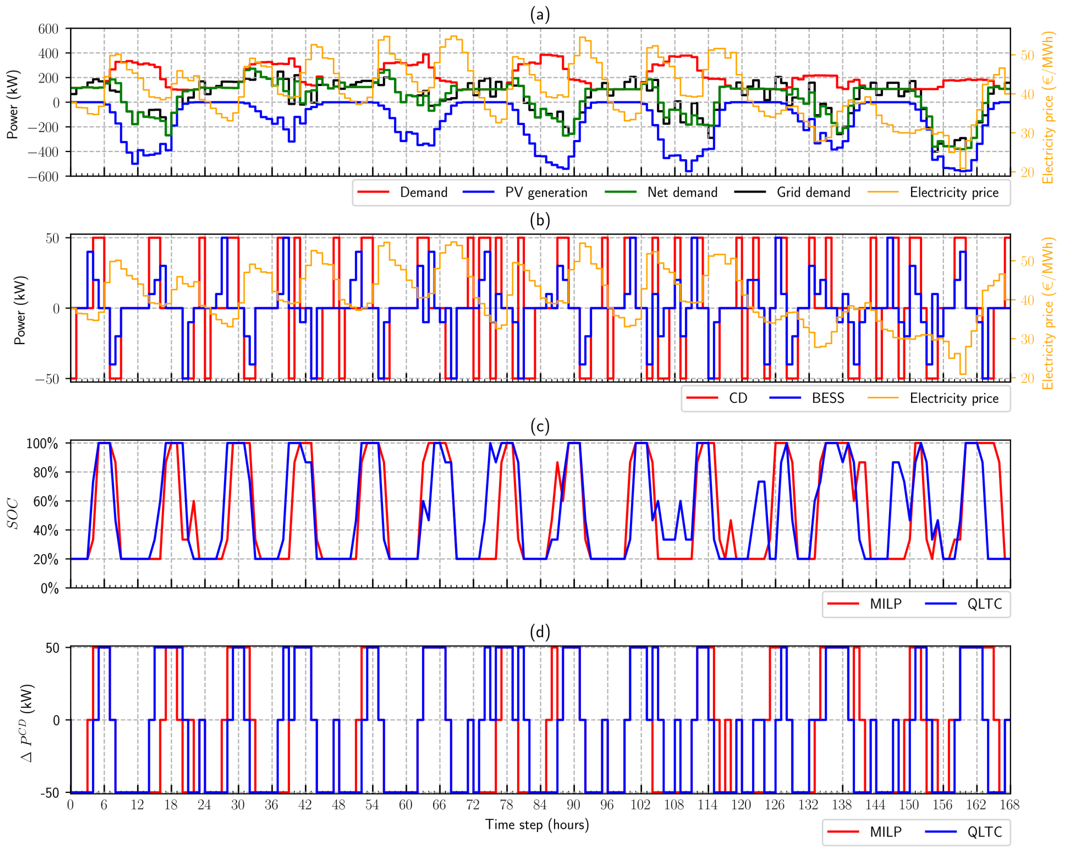


Fig. 6: Operation on the test set. (a) Power levels, and price. (b) Power BESS, CD, and price. (c) BESS's SOC. (d) CD shifted.

close to the MILP's optimum on data it has not encountered before.

References

[1] M. Manic, K. Amarasinghe, J. J. Rodriguez-Andina, and C. Rieger, "Intelligent buildings of the future: Cyberaware, deep learning powered, and human interacting," *IEEE Ind. Electron. Mag.*, vol. 10, no. 4, pp. 32–49, 2016.

[2] G. Tsaousoglou, J. S. Giraldo, and N. G. Paterakis, "Market mechanisms for local electricity markets: A review of models, solution concepts and algorithmic techniques," *Renew. Sust. Energ. Rev.*, vol. 156, p. 111890, 2022.

[3] G. May, B. Stahl, M. Taisch, and D. Kiritsis, "Energy management in manufacturing: From literature review to a conceptual framework," *J. Clean. Prod.*, vol. 167, pp. 1464–1489, 11 2017.

[4] P. P. Vergara, J. C. Lopez, L. C. da Silva, and M. J. Rider, "Security-constrained optimal energy management system for three-phase residential microgrids," *Electr. Pow. Syst. Res.*, vol. 146, pp. 371–382, May 2017.

[5] P. P. Vergara, M. Salazar, J. S. Giraldo, and P. Palensky, "Optimal dispatch of PV inverters in unbalanced distribution systems using reinforcement learning," *Int. J. Electr. Power Energy Syst.*, vol. 136, p. 107628, 2022.

[6] D. Cao, W. Hu, J. Zhao, G. Zhang, B. Zhang, Z. Liu, Z. Chen, and F. Blaabjerg, "Reinforcement learning and its applications in modern power and energy systems: A review," *J. Mod. Power Syst. Clean Energy*, vol. 8, no. 6, pp. 1029–1042, 2020.

[7] E. M. Salazar Duque, J. S. Giraldo, P. P. Vergara, P. Nguyen, A. van der Molen, and H. Slootweg, "Community energy storage operation via reinforcement learning with eligibility traces," *Electric Power Systems Research*, vol. 212, p. 108515, 2022.

[8] Z. Liang, D. Bian, C. Xu, X. Zhang, D. Shi, and Z. Wang, "Deep reinforcement learning based energy management strategy for commercial buildings considering comprehensive comfort levels," *52nd N. Am. Power Symp. (NAPS)*, 2020.

[9] D. J. Harrold, J. Cao, and Z. Fan, "Data-driven battery operation for energy arbitrage using rainbow deep reinforcement learning," *Energy*, vol. 238, p. 121958, 2022.

[10] V.-H. Bui, A. Hussain, and H.-M. Kim, "Double deep *q*-learning-based distributed operation of battery energy storage system considering uncertainties," *IEEE Trans. Smart Grid*, vol. 11, no. 1, pp. 457–469, 2019.

[11] J. Cao, D. Harrold, Z. Fan, T. Morstyn, D. Healey, and K. Li, "Deep reinforcement learning-based energy storage arbitrage with accurate lithium-ion battery degradation model," *IEEE Trans. Smart Grid*, vol. 11, no. 5, pp. 4513–4521, 2020.

[12] C. Hau, K. K. Radhakrishnan, J. Siu, and S. K. Panda, "Reinforcement learning based energy management algorithm for energy trading and contingency reserve application in a microgrid," in *2020 IEEE PES Innov. Smart Grid Technol. Eur. (ISGT-Europe)*. IEEE, 2020, pp. 1005–1009.

[13] E. M. Salazar Duque, J. S. Giraldo, P. P. Vergara, P. Nguyen, A. van der Molen, and H. Slootweg, "Community energy storage operation via reinforcement learning with eligibility traces," *Electric Power Systems Research*, vol. 212, p. 108515, 2022.

[14] S. Zhang and R. S. Sutton, "A deeper look at experience replay," *arXiv preprint arXiv:1712.01275*, 2017.

[15] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[16] A. Perera and P. Kamalaruban, "Applications of reinforcement learning in energy systems," *Renew. Sust. Energ. Rev.*, vol. 137, p. 110618, 2021.